高职高专计算机任务驱动模式教材

ASP.NET动态网站开发

李利杰　主　编　　　张颖　蒋宁　副主编



清华大学出版社

高职高专计算机任务驱动模式教材

ASP.NET 动态网站开发

李利杰　主　编
张　颖　蒋　宁　副主编

清华大学出版社
北　京

内 容 简 介

本书通过完整的、极具代表性的电子商城项目开发的全过程，系统介绍了使用 ASP. NET 开发动态网站所使用的各类主流技术。全书围绕 Smart On Line 电子商城的项目演示及数据库设计、网站页面开发、会员注册、会员登录、商品展示、会员购物、商品信息管理、订单管理、网站发布的实现过程展开，将知识介绍和技能训练进行了有机结合。

本书可作为高校计算机、电子商务及信息类相关专业的教学用书，也可供有关领域的各类培训班、计算机从业人员和 Web 应用开发爱好者参考使用。

图书在版编目(CIP)数据

ASP. NET 动态网站开发/李利杰主编. —北京：清华大学出版社，2015
高职高专计算机任务驱动模式教材
ISBN 978-7-302-37971-3

Ⅰ. ①A…　Ⅱ. ①李…　Ⅲ. ①网页制作工具—程序设计—高等职业教育—教材　Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字(2014)第 209480 号

责任编辑：张龙卿
封面设计：徐日强
责任校对：李　梅
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795764
印 装 者：北京嘉实印刷有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　张**：19.5　　**字　　数**：469 千字
版　　次：2015 年 1 月第 1 版　　**印　　次**：2015 年 1 月第1 次印刷
印　　数：1～2500
定　　价：38.00 元

产品编号：059924-01

编审委员会

出版说明

我国高职高专教育经过十几年的发展，已经转向深度教学改革阶段。教育部于2006年12月发布了教高[2006]第16号文件《关于全面提高高等职业教育教学质量的若干意见》，大力推行工学结合，突出实践能力培养，全面提高高职高专教学质量。

清华大学出版社作为国内大学出版社的领跑者，为了进一步推动高职高专计算机专业教材的建设工作，适应高职高专院校计算机类人才培养的发展趋势，根据教高[2006]第16号文件的精神，2007年秋季开始了切合新一轮教学改革的教材建设工作。该系列教材一经推出，就得到了很多高职院校的认可和选用，其中部分书籍的销售量都超过了3万册。现重新组织优秀作者对部分图书进行改版，并增加了一些新的图书品种。

目前国内高职高专院校计算机网络与软件专业的教材品种繁多，但符合国家计算机网络与软件技术专业领域技能型紧缺人才培养培训方案，并符合企业的实际需要，能够自成体系的教材还不多。

我们组织国内对计算机网络和软件人才培养模式有研究并且有过一段实践经验的高职高专院校，进行了较长时间的研讨和调研，遴选出一批富有工程实践经验和教学经验的双师型教师，合力编写了这套适用于高职高专计算机网络、软件专业的教材。

本套教材的编写方法是以任务驱动、案例教学为核心，以项目开发为主线。我们研究分析了国内外先进职业教育的培训模式、教学方法和教材特色，消化吸收优秀的经验和成果。以培养技术应用型人才为目标，以企业对人才的需要为依据，把软件工程和项目管理的思想完全融入教材体系，将基本技能培养和主流技术相结合，课程设置中重点突出、主辅分明、结构合理、衔接紧凑。教材侧重培养学生的实战操作能力，学、思、练相结合，旨在通过项目实践，增强学生的职业能力，使知识从书本中释放并转化为专业技能。

一、教材编写思想

本套教材以案例为中心，以技能培养为目标，围绕开发项目所用到的知识点进行讲解，对某些知识点附上相关的例题，以帮助读者理解，进而将知识转变为技能。

考虑到是以“项目设计”为核心组织教学，所以在每一学期配有相应的实训课程及项目开发手册，要求学生在教师的指导下，能整合本学期所学的知识内容，相互协作，综合应用该学期的知识进行项目开发。同时，在教材中采用了大量的案例，这些案例紧密地结合教材中的各个知识点，循序渐进，由浅入深，在整体上体现了内容主导、实例解析、以点带面的模式，配合课程后期以项目设计贯穿教学内容的教学模式。

软件开发技术具有种类繁多、更新速度快的特点。本套教材在介绍软件开发主流技术的同时，帮助学生建立软件相关技术的横向及纵向的关系，培养学生综合应用所学知识的能力。

二、丛书特色

本系列教材体现目前工学结合的教改思想，充分结合教改现状，突出项目面向教学和任务驱动模式教学改革成果，打造立体化精品教材。

（1）参照和吸纳国内外优秀计算机网络、软件专业教材的编写思想，采用本土化的实际项目或者任务，以保证其有更强的实用性，并与理论内容有很强的关联性。

（2）准确把握高职高专软件专业人才的培养目标和特点。

（3）充分调查研究国内软件企业，确定了基于Java和.NET的两个主流技术路线，再将其组合成相应的课程链。

（4）教材通过一个个的教学任务或者教学项目，在做中学，在学中做，以及边学边做，重点突出技能培养。在突出技能培养的同时，还介绍解决思路和方法，培养学生未来在就业岗位上的终身学习能力。

（5）借鉴或采用项目驱动的教学方法和考核制度，突出计算机网络、软件人才培训的先进性、工具性、实践性和应用性。

（6）以案例为中心，以能力培养为目标，并以实际工作的例子引入概念，符合学生的认知规律。语言简洁明了、清晰易懂，更具人性化。

（7）符合国家计算机网络、软件人才的培养目标；采用引入知识点、讲述知识点、强化知识点、应用知识点、综合知识点的模式，由浅入深地展开对技术内容的讲述。

（8）为了便于教师授课和学生学习，清华大学出版社正在建设本套教材的教学服务资源。在清华大学出版社网站（www.tup.com.cn）免费提供教材的电子课件、案例库等资源。

高职高专教育正处于新一轮教学深度改革时期，从专业设置、课程体系建设到教材建设，依然是新课题。希望各高职高专院校在教学实践中积极提出意见和建议，并及时反馈给我们。清华大学出版社将对已出版的教材不断地修订、完善，提高教材质量，完善教材服务体系，为我国的高职高专教育继续出版优秀的高质量的教材。

清华大学出版社

高职高专计算机任务驱动模式教材编审委员会

2014年3月

前　言

动态网站开发是计算机应用技术、电子商务和信息管理专业的一门重要的专业必修课，也是一门专业核心课程。目前国内对网页制作及网站开发人员的需求量正急速增加，现今社会十分缺乏优秀的网站设计与开发人员，网站开发工程师拥有很好的职业前景。

本书以完整的行业项目为载体，采用课上课下两个项目并进的形式，固化学生的实践技能；教材建设紧跟行业需求，内容覆盖 CSS、JavaScript、局部刷新、数据库存储过程和 Web Service。本书使用大量的图例和源代码帮助零起点的人员迅速掌握网站开发的核心技术。

本书一共有 9 章，分为 5 个部分。

第一部分包含项目 1，讲解 Smart On Line 电子商城功能和相应 Smart 数据库的设计和实现方法，读者将从中学会 SQL Server 2008 数据库操作和常用的 SQL 语法，如创建表、设置外键约束等。

第二部分包含项目 2～项目 4，主要讲解 Smart On Line 电子商城的首页制作、用户注册和登录功能的实现方法，将学习 CSS、JavaScript、ASP.NET 常见 Web 服务器控件、数据验证控件、内置对象和 ADO.NET 六大内容。

第三部分包含项目 5 和项目 6，主要讲解商品展示、会员购物功能的具体实现方法，读者从中可掌握 GridView 和 DataList 常规操作和高级应用技巧，强化应用数据库存储过程、事务等高级技术。

第四部分包含项目 7 和项目 8，重点讲解商品信息管理和订单管理的实现方法，通过这两个项目的训练，读者可以进一步掌握 GridView 模板列编辑和在数据库中保存图片等实战技术，并接触先进的行业技术，包括 Web Service、邮件发送等技术。

第五部分只包含项目 9，该项目讲解了 Smart On Line 电子商城的发布过程，通过这个项目的训练，读者可以掌握 web.config 配置文件加解密和应用 IIS 发布 ASP.NET 应用程序的基本过程。

本书提供了完整的配套教学资料和课程网站，包括实例源代码、PPT 电子课件和授课录像，为读者提供全方位的学习环境。在每一个项目后面均配有一定量的习题，读者可以通过这些习题对所学知识进行巩固，加深理

解，并学会在项目中运用所学知识来解决实际问题的能力。需要强调的一点是，对于每个项目后的同步项目操练，一定要亲自上机实践，只有多上机才能发现问题、解决问题，才能取得事半功倍的学习效果。

本书编写过程中得到众多同事和朋友的帮助，在此向所有帮助和支持过我们的朋友表示感谢。

编　者

2014年10月

目　录

项目 1　数据库设计及创建

1.1　概　　述

ASP. NET 是统一的 Web 应用程序平台，提供为建立和部署企业级 Web 应用程序所必需的服务。ASP. NET 为能够面向任何浏览器或设备建立更安全的、更强的可升级性、更稳定的应用程序提供了新的编程模型和基础结构。同时 ASP. NET 是 Microsoft .NET Framework 的一部分，是一种可以在高度分布的 Internet 环境中简化应用程序开发的计算环境。.NET Framework 包含公共语言运行库，它提供了各种核心服务，如内存管理、线程管理和代码安全。它也包含.NET Framework 类库，这是一个开发人员用于创建应用程序的综合的、面向对象的类型集合。本书以一个完整的电子商城网站(Smart On Line 电子商城)为载体，通过对 Smart On Line 电子商城开发的全程讲解和学习，使读者掌握 ASP. NET的关键技术，从而能够胜任使用 ASP. NET 技术开发动态网站的工作岗位。本书将 Smart On Line 电子商城整个项目分解为 9 个子项目进行相对独立的讲解。

1.2　Smart On Line 电子商城项目介绍

Smart On Line 电子商城是为各类人士提供休闲、便捷的商品交易平台。Smart On Line 电子商城的用户角色分为注册用户、匿名用户和管理员，因此本系统的开发围绕这三种不同的角色进行开发，以满足系统的需求。

该项目主要实现如下功能。

- 匿名用户：通过本网站了解各种商品的信息，通过在线注册成为会员。
- 注册用户：通过本网站了解商品的信息，购买商品，下订单，发表评论等。
- 管理员：通过本系统管理和维护站点的基本运行、维护商品目录信息、商品信息、订单信息等。

Smart On Line 电子商城的主要功能如图 1-1 所示。



图 1-1　Smart On Line 电子商城的功能

下面对每个功能进行详细的讲解。

(1) 搜索商品

提供的强大的模糊查询功能,以便快捷地找到用户感兴趣的商品。

(2) 会员注册

电子商城允许浏览者在线填写注册表并实时成为商城会员,注册表中采用验证码技术来防止恶意注册。为防止用户注册的敏感信息在传输过程中被窃取,采用加密算法对注册用户的部分重要信息进行加密。支持忘记密码功能,会员可以通过此功能查找忘记的密码。

(3) 会员登录

用户购买商品前需进行用户登录。会员登录功能需支持会员当前在线标示、会员登录唯一性检验、会员登录日志信息保存功能。登录成功后可随时查看账务明细、订单明细。

(4) 商品展示

允许用户选择网格和列表两种不同的商品展示方式以适合客户的最佳观看效果。商品的信息栏中介绍商品外观、商品性能、商品参数、注意事项、优惠信息、配送信息等。商品信息栏中支持加入购物车和查看详细商品信息功能。

(5) 会员购物

引导会员完成购物流程。首先是显示会员选购的商品明细,可以更改购买数量,删除不需要的商品,重新计算价格等;然后是填写订单内容,如收货人、地址、配送方式(配送方式可由用户从商城管理系统中进行定义,如配送方式名称、价格等)、付款方式等。

(6) 商品分类管理

商品分类管理用来设置商品分类,分类可以是多级。例如,图书类商品的分类可能是这样:首先是图书类,图书类下面又分为文学类、艺术类、计算机类等,计算机类下面又分为图像处理、应用软件、程序开发等。针对每一种分类,都可以定义它的分类介绍类型。例如,针对图书,可能的分类介绍类型有"内容简介"、"目录"、"作者介绍"、"书评"等。针对家电产品,可能的分类介绍类型有"规格"、"功能说明"等。

(7) 商品信息管理

商品信息管理用来维护商品信息,如商品名称、描述、商品图片、简介等。

(8) 订单管理

订单管理主要完成用户订单的相关功能,支持订单列表——显示当前会员的订单,如订单号、消费金额、运费、状态等。单击后可查看订单明细。支持订单明细——查看选定的订单中,所订购商品的详细情况,包括商品名称、数量、单价等。支持订单查询——根据订单号、日期、状态等信息查询相关的订单。

(9) 用户管理

用户管理功能支持管理员按照不同条件检索会员;支持管理员通过后台查看或修改会员信息;支持查看会员登录电子商城网站等日志信息。

1.3 项目1分析

业界开发动态网站的技术常见的有ASP.NET、JSP、PHP。鉴于本项目组以前具备采用C#语言开发桌面程序的经验,项目组成员经过讨论,决定采用ASP.NET技术以减小项目开发风险。项目组同时也决定采用SQL Server 2008数据库管理系统进行Smart On

Line 电子商城的数据存储。所用开发技术确定后，项目组成员所要做的第一步工作就是进行 Smart On Line 电子商城的数据库设计及创建，首先创建数据库关系图，接着对基本表中各字段进行细分、定义，形成数据库逻辑模式；然后根据用户处理的要求、安全性的考虑，在基本表的基础上建立必要的视图、约束、存储过程、触发器等。

1.4　知识准备

1.4.1　SQL Server 2008 概述

Microsoft SQL Server 2008 提供完整的企业级技术和工具，以最低的成本获得最有价值的信息。具有高性能、高可用性、高安全性等特性，使用更多的高效管理与开发工具，并利用自带的商业智能工具实现更为广泛深入的商业开发。SQL Server 2008 R2 以.NET Framework 3.5 SP1、Visual Studio 2008、BizTalk Server（企业商务应用）、Office 为基础支撑，包含关系数据库（RDBMS）、层次类型数据（XML）、联机分析处理（OLAP）和文件流（FileStream）在内的数据管理平台，提供了查询、检索、集成、报表和分析等高效工具，广泛应用于基于移动设备和台式设备的分布式环境数据库应用，如图 1-2 所示。

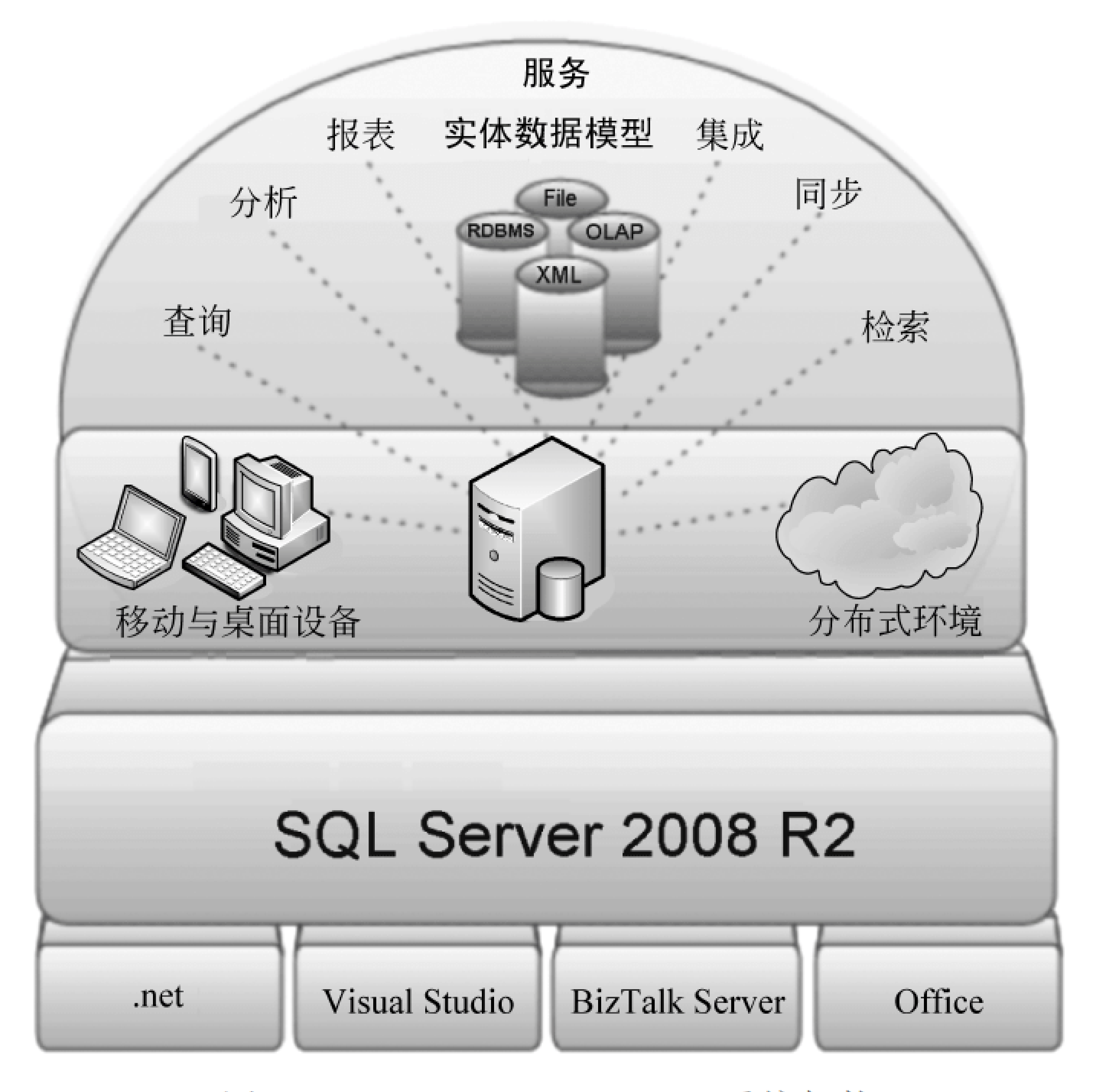


图 1-2　SQL Server 2008 R2 系统架构

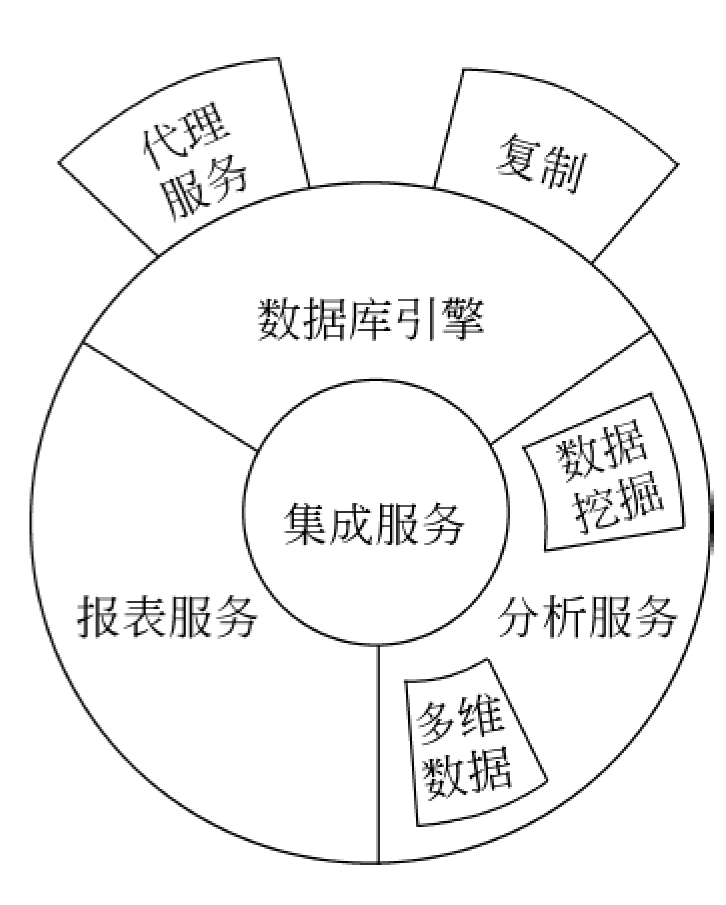


图 1-3　服务器组件

SQL Server 2008 R2 服务器由数据库引擎（基础）、集成服务、分析服务和报表服务 4 个服务器组件构成，如图 1-3 所示。

（1）数据库引擎：存储、处理和保护数据（关系、XML 数据）的核心服务、复制、全文搜索等。

（2）分析服务（Analysis Services）：创建和管理联机分析处理（OLAP）以及数据挖掘的

工具，为商业智能应用程序提供了联机分析处理(OLAP)和数据挖掘功能。分析服务允许开发人员设计、创建和管理包含从其他数据源(如关系数据库)聚合的数据的多维结构，以实现对OLAP的支持。对于数据挖掘应用程序，分析服务允许开发人员设计、创建和可视化处理那些通过使用各种行业标准数据的挖掘算法，以及根据其他数据源构造出来的数据挖掘模型。

(3) 报表服务(Reporting Services)：创建、管理和部署报表(表格、矩阵、图形和自由格式)的服务器和客户端组件，是基于服务器的报表平台，为各种数据源提供了完善的报表功能。Reporting Services包含一整套可用于创建、管理和传送报表的工具以及允许开发人员在自定义应用程序中集成或扩展数据和报表处理的应用程序编程接口(Application Programming Interface，API)。Reporting Services工具在Microsoft Visual Studio环境中工作，并与SQL Server工具和组件完全集成。

(4) 集成服务(Integration Services)：用于移动、复制和转换数据的图形工具和可编程对象，是用于数据集成和数据转换解决方案的平台，包含用于生成和调试包的图形工具和向导；用于执行工作流功能的任务，例如FTP(File Transfer Protocol，文件传输协议)操作、SQL语句执行和电子邮件消息处理；用于提取和加载数据的数据源和目标；用于清理、聚合、合并和复制数据的转换；用于管理集成服务包的集成服务；对Integration Services对象模型基于应用程序编程接口(API)进行编程。设计Integration Services项目的主要任务就是定义控制流、数据流及事件处理程序。

1.4.2 SQL Server 2008 的安装

安装一个SQL Server 2008 R2数据库服务器，在向导的帮助下，基本上是选择默认值和单击“下一步”按钮，十分简单。但真正理解SQL Server安装选项的含义要到学完本书时才能有较深的认识。真正安装一个能够承担成千上万人同时访问的SQL Server 2008 R2数据库服务器，还需要学习和查询很多知识。本节简单地介绍SQL Server 2008 R2安装的入门知识。

(1) 以Administrator Windows管理员登录服务器，从安装介质中运行SQL Server 2008 R2的安装程序setup.exe，开始安装。

(2) 在“SQL Server安装中心”界面中选择“安装”选项，单击“全新安装或向现有安装添加功能”，如图1-4所示。

(3) 在“安装程序支持规则”界面，Windows运行环境检测通过后，单击“确定”按钮。

(4) 在“产品密钥”界面，输入产品密钥，单击“下一步”按钮。

(5) 在“许可条款”界面，选中“我接收许可条款”选项，单击“下一步”按钮。

(6) 在“安装程序支持文件”界面，单击“安装”按钮。

(7) 在“安装程序支持规则”界面，操作完成后，已通过9项、失败0项，单击“下一步”按钮。

(8) 在“设置角色”界面，选择“SQL Server功能安装”选项，单击“下一步”按钮。

(9) 在“功能选择”界面，根据实际情况选择相应的组件，如果不知道需要安装哪些组件，可以单击“全选”按钮，设置或记录“共享功能目录”，单击“下一步”按钮。

(10) 在“安装规则”界面，规则检查通过了6项，失败了0项，单击“下一步”按钮。

(11) 在“安装配置”界面，选择“默认实例”选项，设置并记录“实例ID”和“实例根目

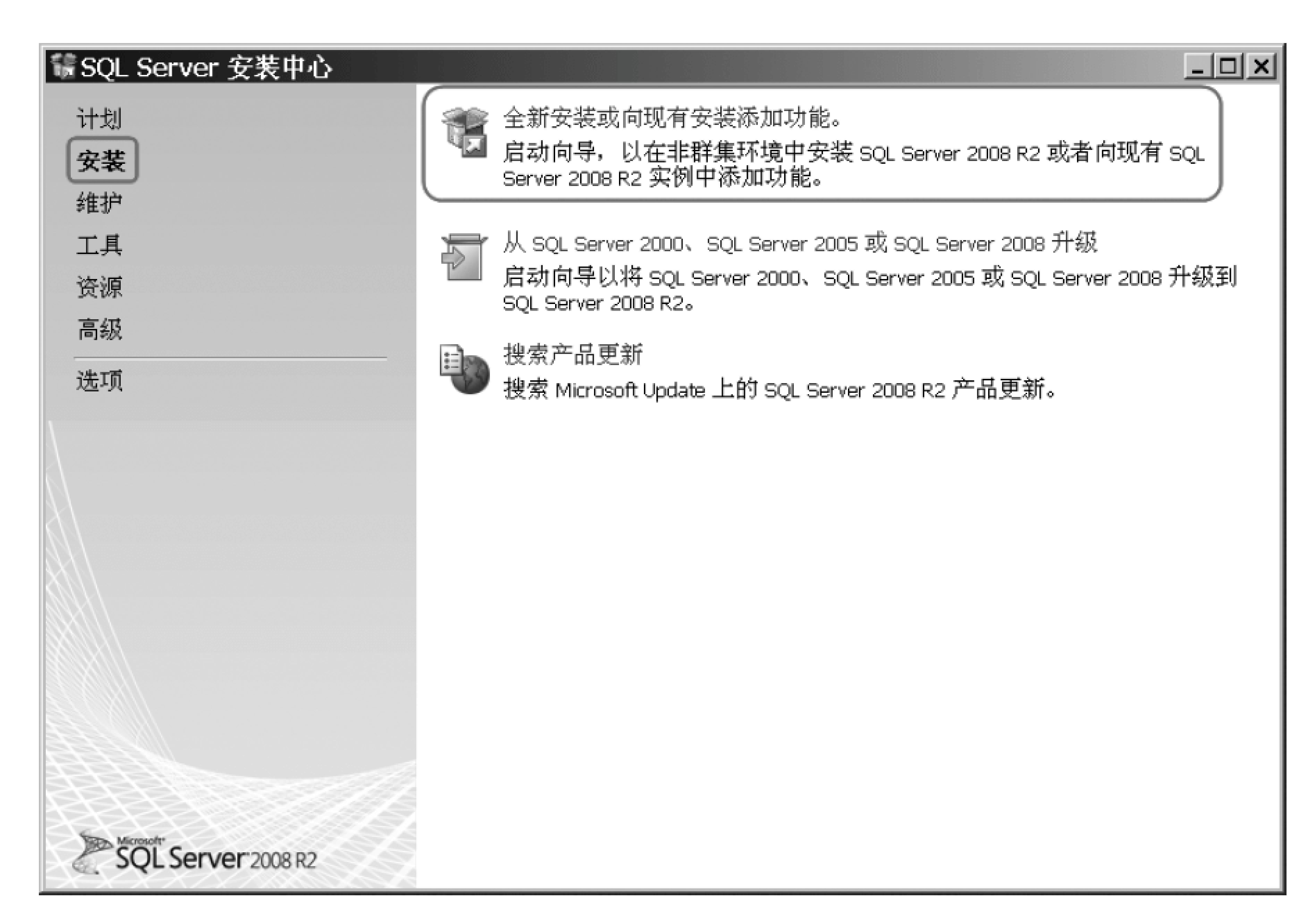


图 1-4　“SQL Server 安装中心”中的“安装”界面

录”，单击“下一步”按钮。

(12) 在“磁盘空间要求”界面，显示安装所需的磁盘空间，单击“下一步”按钮。

(13) 在“服务器配置”界面，显示服务账户，单击“下一步”按钮。保证 SQL Server Database Engine 服务是 SYSTEM 权限，否则 SQL Server 服务器无法启动，单击“下一步”按钮，如图 1-5 所示。



图 1-5　“服务器配置”界面

(14) 在“数据库引擎配置”界面，根据需求选择身份验证模式，建议“身份验证模式”选项组中选择“混合模式”，设置并记录下 sa 密码(不要设置过于简单，不然服务器很容易被黑)，单击“添加当前用户”按钮，将当前 Windows 登录用户指定为 SQL Server 管理员，单击“下一步”按钮，如图 1-6 所示。

(15) 在“Reporting Services 配置”界面，单击“添加当前用户”按钮，可以管理 Analysis



图 1-6 “数据库引擎配置”界面

服务，单击“下一步”按钮。

（16）在“Reporting Services 配置”界面，选中“安装本机模式默认配置”选项，单击“下一步”按钮。

（17）在“错误报告”界面，单击“下一步”按钮。

（18）在“安装配置规则”界面，显示操作完成，已通过 6 项，失败 0 项，单击“下一步”按钮。

（19）在“准备安装”界面，单击“安装”按钮，在“准备进度”界面中会显示安装进度，请等待安装结果。

（20）安装完成后，在“完成”界面中会显示“SQL Server 2008 R2 安装已成功完成”的提示信息，单击“关闭”按钮。

1.4.3 SQL Server 2008 的配置

SQL Server 2008 R2 端口配置和服务的操作步骤如下。

（1）在安装 SQL Server 2008 R2 软件的计算机上，单击“开始”→“程序”→“Microsoft SQL Server 2008 R2”→“配置工具”→“SQL Server 配置管理器”命令，弹出“SQL Server 配置管理器”窗口，如图 1-7 所示。

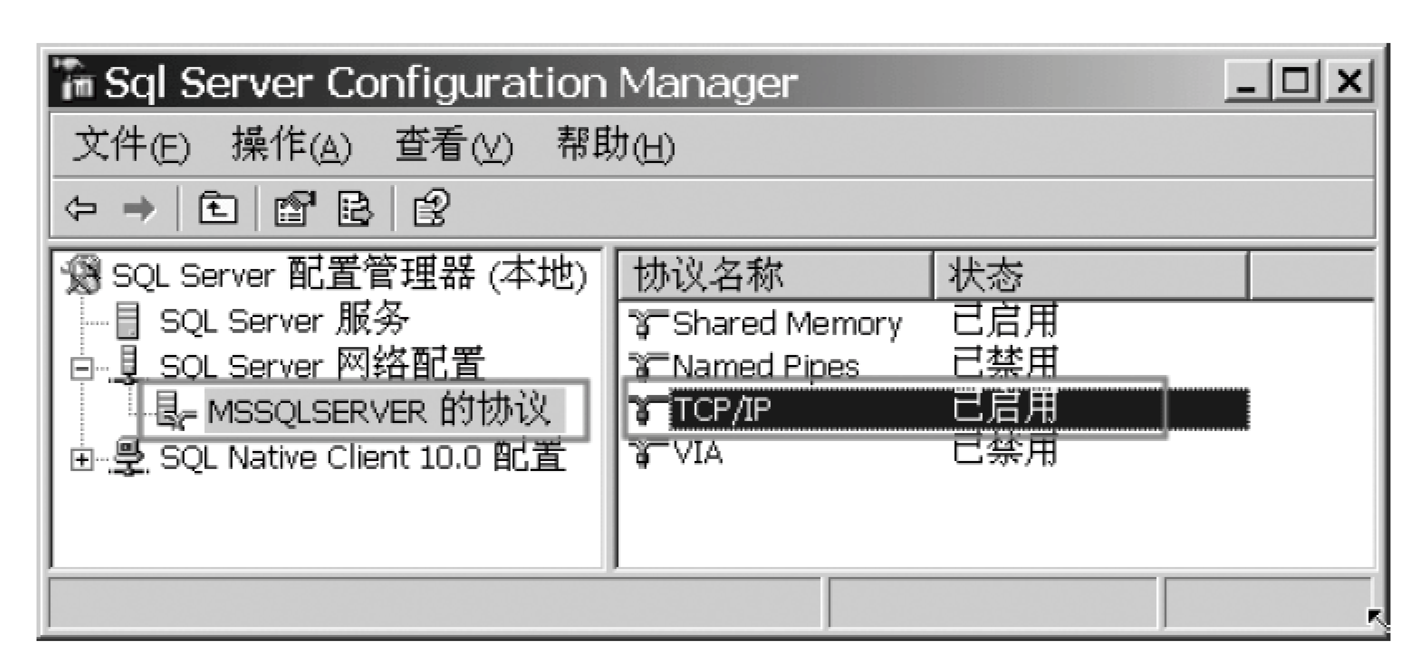


图 1-7 “SQL Server 配置管理器”界面

(2) 在"SQL Server 配置管理器"窗口,单击"MSSQLSERVER 的协议",查看 TCP/IP 的状态是否是已启用,双击 TCP/IP,弹出"TCP/IP 属性"窗口,如图 1-8 所示。

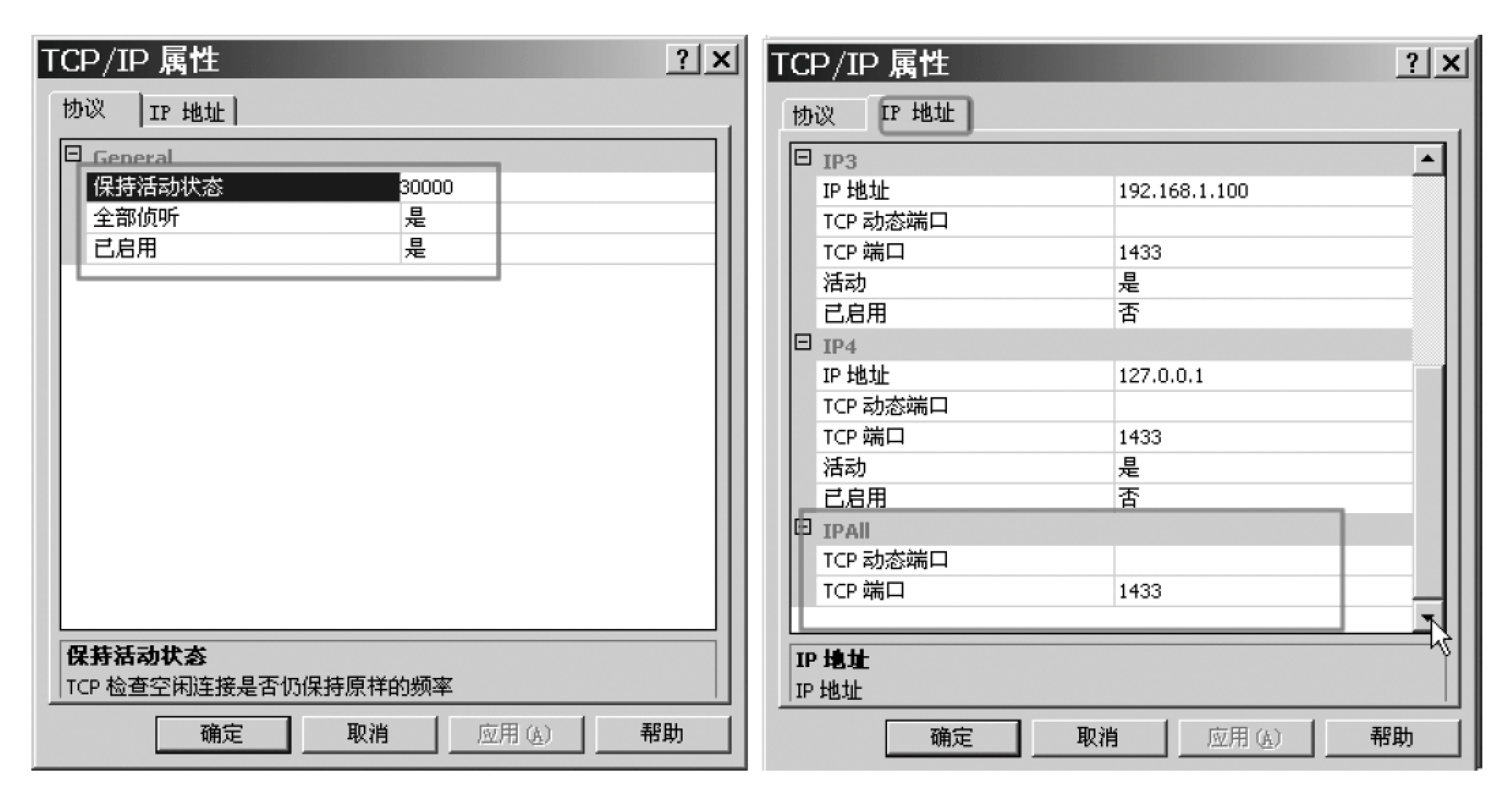
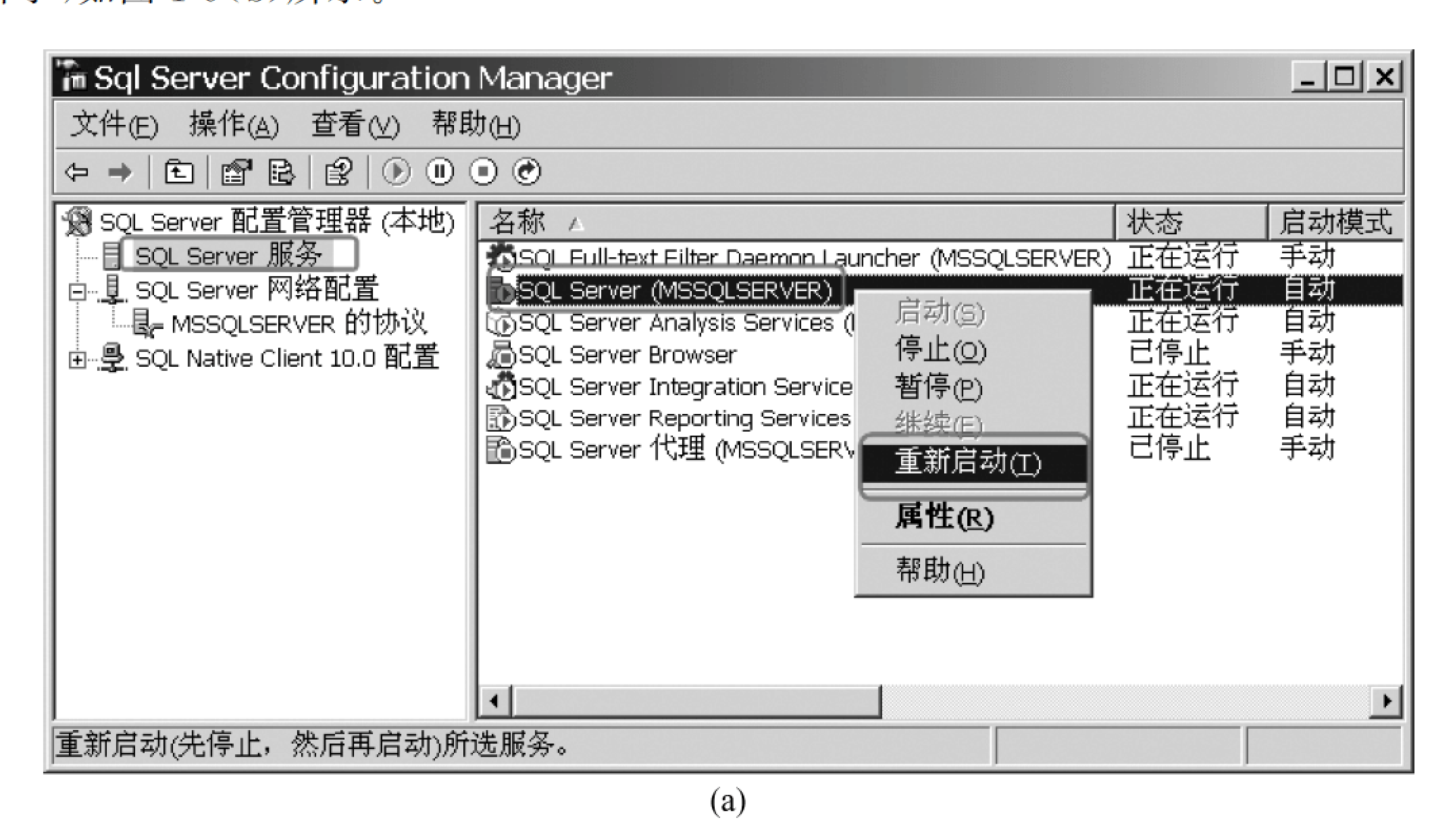


图 1-8　"TCP/IP 属性"界面

(3) 在"TCP/IP 属性"窗口中单击"IP 地址",保证 IPALL 里面的 TCP 端口是 1433,单击"确定"按钮返回。

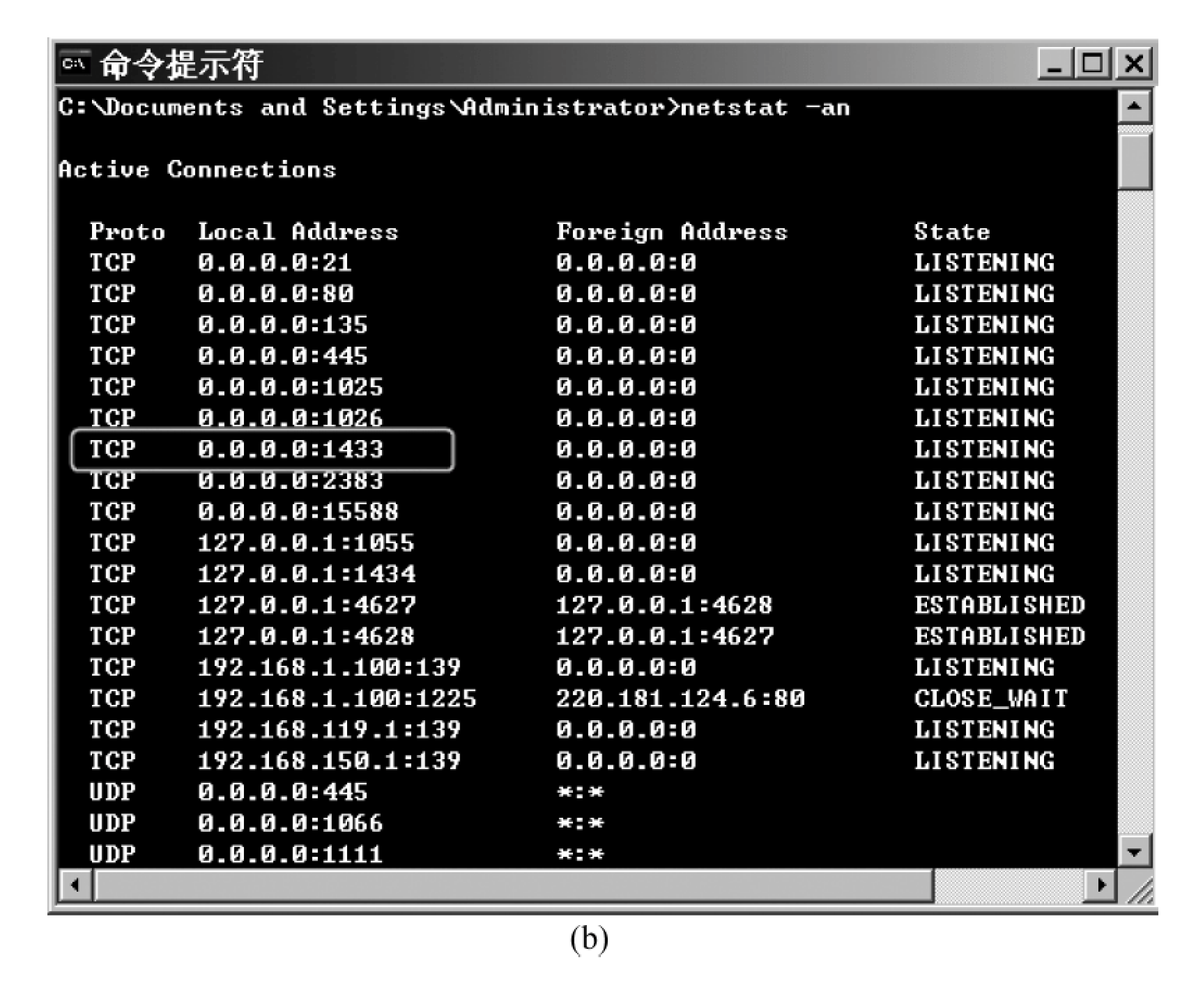
(4) 在"SQL Server 配置管理器"窗口,单击"SQL Server 服务",右击 SQL Server (MSSQLSERVER),单击"重新启动"命令,如图 1-9(a)所示。然后关闭"SQL Server 配置管理器"。

(5) 在命令行下输入 netstat-an,如果找到"0.0.0.0:1433",就说明 SQL Server 开始监听了,如图 1-9(b)所示。



(a)

图 1-9　重启"SQL Server 配置管理器"和查看端口监听



(b)

图 1-9(续)

(6)在桌面上右击“我的电脑”,单击“管理”命令,弹出“计算机管理”窗口,如图 1-10 所示,在左侧窗格中单击“服务”可显示相应的服务。图中标示部分为 SQL Server 服务列表。



图 1-10 SQL Server 2008 服务列表

1.4.4 SQL Server 集成管理器的使用

SQL Server Management Studio(SSMS,SQL Server 集成管理器,可简写为

Management Studio)是为 SQL Server 数据库管理员和开发人员提供的新工具。此工具由 Visual Studio 内部承载,它提供了用于数据库管理的图形工具和功能丰富的开发环境。Management Studio 是一个功能强大且灵活的工具。但是,初次使用 Visual Studio 的用户有时无法以最快的方式访问所需的功能。下面介绍 Management Studio 的基本使用方法。

1. 启动 Management Studio

在“开始”菜单上,依次选择“所有程序”→Microsoft SQL Server 2008,再单击 SQL Server Management Studio。首先弹出“连接到服务器”对话框(图 1-11)。在“连接到服务器”对话框中采用默认设置(Windows 身份验证),再单击“连接”按钮。默认情况下,Management Studio 中将显示三个组件窗口,如图 1-12 所示。



图 1-11 打开时的 SQL Server Management Studio

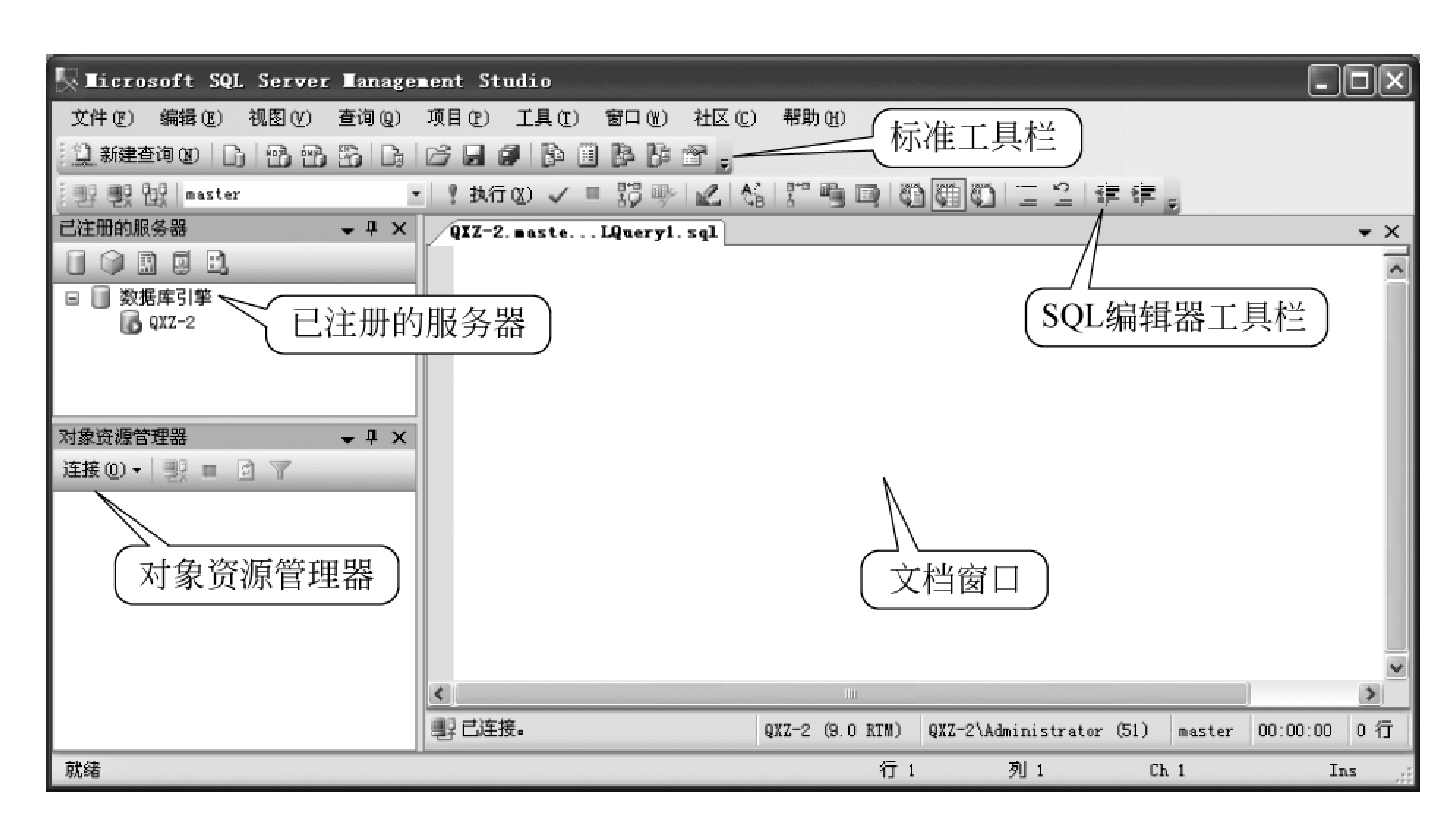


图 1-12 SQL Server Management Studio 的窗体布局

“已注册的服务器”窗口列出的是经常管理的服务器。可以在此列表中添加和删除服务器。对象资源管理器是服务器中所有数据库对象的树视图。此树视图可以包括 SQL Server Database Engine、Analysis Services、Reporting Services、Integration Services 和 SQL Server Mobile 的数据库。对象资源管理器包括与其连接的所有服务器的信息。打开 Management Studio 时，系统会提示用户将对象资源管理器连接到上次使用的设置。可以在“已注册的服务器”组件中双击任意服务器进行连接，或在任意服务器上右击并选择“连接”→“对象资源管理器”命令。而要连接的服务器是无须再注册的。文档窗口是 Management Studio 中的最大部分。文档窗口可能包含查询编辑器和浏览器窗口。默认情况下，将显示已与当前计算机上的数据库引擎实例连接的“摘要”页。

2. 与已注册的服务器和对象资源管理器连接

(1) 连接到服务器。已注册的服务器组件的工具栏包含用于数据库引擎、Analysis Services、Reporting Services、SQL Server Mobile 和 Integration Services 的按钮。注册 AdventureWorks 数据库(下载地址：http://www.codeplex.com/SqlServerSamples)分为以下 5 个步骤。

① 如有必要，单击“已注册的服务器”工具栏上的“数据库引擎”选项(该选项可能已选中)。

② 右击“数据库引擎”选项，选择“新建服务器注册”命令，此时将打开“新建服务器注册”对话框。

③ 在“服务器名称”文本框中，输入 SQL Server 实例的名称。

④ 在“已注册的服务器名称”文本框中，输入 AdventureWorks。

⑤ 在“连接属性”选项卡的“连接到数据库”列表中，选择 AdventureWorks，再单击“保存”按钮。

以上操作说明可以更改默认的服务器名称。

(2) 与对象资源管理器连接。与已注册的服务器类似，对象资源管理器也可以连接到数据库引擎、Analysis Services、Integration Services、Reporting Services 和 SQL Server Mobile。方法如下。

① 在对象资源管理器的工具栏上，单击“连接”，显示可用连接类型下拉列表，再选择“数据库引擎”，系统将打开“连接到服务器”对话框。

② 在“服务器名称”文本框中输入 SQL Server 实例的名称。

③ 单击“选项”，然后浏览各个选项。

④ 单击“连接”，连接到服务器。如果已经连接，则将直接返回到对象资源管理器，并将该服务器设置为焦点。

⑤ 在对象资源管理器中展开“数据库”文件夹，然后选择 AdventureWorks。

3. 连接查询编辑器

Management Studio 是一个集成开发环境，允许开发人员在与服务器断开连接时编写或编辑代码。当服务器不可用或要节省短缺的服务器或网络资源时，这一点很有用。也可以更改查询编辑器与 SQL Server 新实例的连接，而无须打开新的查询编辑器窗口或重新输入代码。

脱机编写代码，然后连接到其他服务器的方法如下。

① 在 Management Studio 工具栏上单击“数据库引擎查询”按钮，以打开查询编辑器。

② 在“连接到数据库引擎”对话框中单击“取消”按钮。系统将打开查询编辑器，同时，查询编辑器的标题栏将提示没有连接到 SQL Server 实例。

③ 在代码窗格中，输入 T-SQL 语句，例如，SELECT * FROM Production. Product。

④ 此时，可以单击“连接”、“执行”、“分析”或“显示估计的执行计划”等按钮以连接到 SQL Server 实例，另外，“查询”菜单、查询编辑器工具栏，或在“查询编辑器”窗口中右击时，显示的快捷菜单中均提供了这些选项。对于本练习，我们将使用工具栏。

⑤ 在工具栏上，单击“执行”按钮，打开“连接到数据库引擎”对话框。

⑥ 在“服务器名称”文本框中输入服务器名称，再单击“选项”按钮。

⑦ 在“连接属性”选项卡上的“连接到数据库”列表中，浏览服务器以选择 AdventureWorks，再单击“连接”按钮。

⑧ 若要使用同一个连接打开另一个“查询编辑器”窗口；可在工具栏上单击“新建查询”选项。

⑨ 若要更改连接，在“查询编辑器”窗口中右击，选择“连接”命令，再单击“更改连接”按钮。

⑩ 在“连接到 SQL Server”对话框中，选择 SQL Server 的另一个实例（如果有），再单击“连接”按钮。

利用查询编辑器的这项新功能，可以在多台服务器上轻松运行相同的代码。这对于涉及类似服务器的维护操作很有效。

程序员通常会问：“我如何才能获得更多的代码编写空间?”有两种方法可以解决此问题，并且都非常简单：一种是最大化查询编辑器窗口；另一种是隐藏不使用的工具窗口。

① 最大化查询编辑器窗口的方法。

单击“查询编辑器”窗口中的任意位置，按 Shift＋Alt＋Enter 组合键，在全屏显示模式和常规显示模式之间进行切换，这种方法适用于任何文档窗口。

② 自动隐藏所有工具窗口的方法。

单击“查询编辑器”窗口中的任意位置。在“窗口”菜单上，选择“自动全部隐藏”命令；若要还原工具窗口，可打开每个工具，再单击窗口上的“自动隐藏”按钮以打开此窗口。

1.4.5　创建数据库

如果开发人员想要访问数据库对象，那么必须正确地确定数据库引用，这是因为数据库对象是其他对象的顶端对象。只有在数据库引用的范围内，其他对象才能被访问。创建数据库需要使用 SQL Server 2008 的数据库引擎服务。数据库引擎是存储、处理、保护数据库和创建数据库的核心。在安装 SQL Server 2008 时，数据库引擎服务注册为 Windows 服务，但是数据库引擎服务不能够直接访问和使用，必须借助前端工具才能连接到数据库引擎。本节分别介绍如何使用 SQL Server Management Studio 图形化方式和 T-SQL 方式创建数据库。

（1）使用 SQL Server Management Studio 以图形化方式创建数据库。

SQL Server 2008 为开发人员提供了图形化方式来创建数据库。下面将介绍使用 SQL Server Management Studio 图形化方式创建数据库的基本步骤。

① 选择桌面中的“开始”→Microsoft SQL Server 2008→SQL Server Management

Studio 命令，打开 SQL Server Management Studio。

② 在打开的“连接到服务器”对话框中输入“登录名”和“密码”，单击“连接”按钮，如图 1-13 所示。



图 1-13 “连接服务器”对话框

③ 选择“视图”菜单中的“对象资源管理器”命令，打开“对象资源管理器”窗口，如图 1-14 所示。

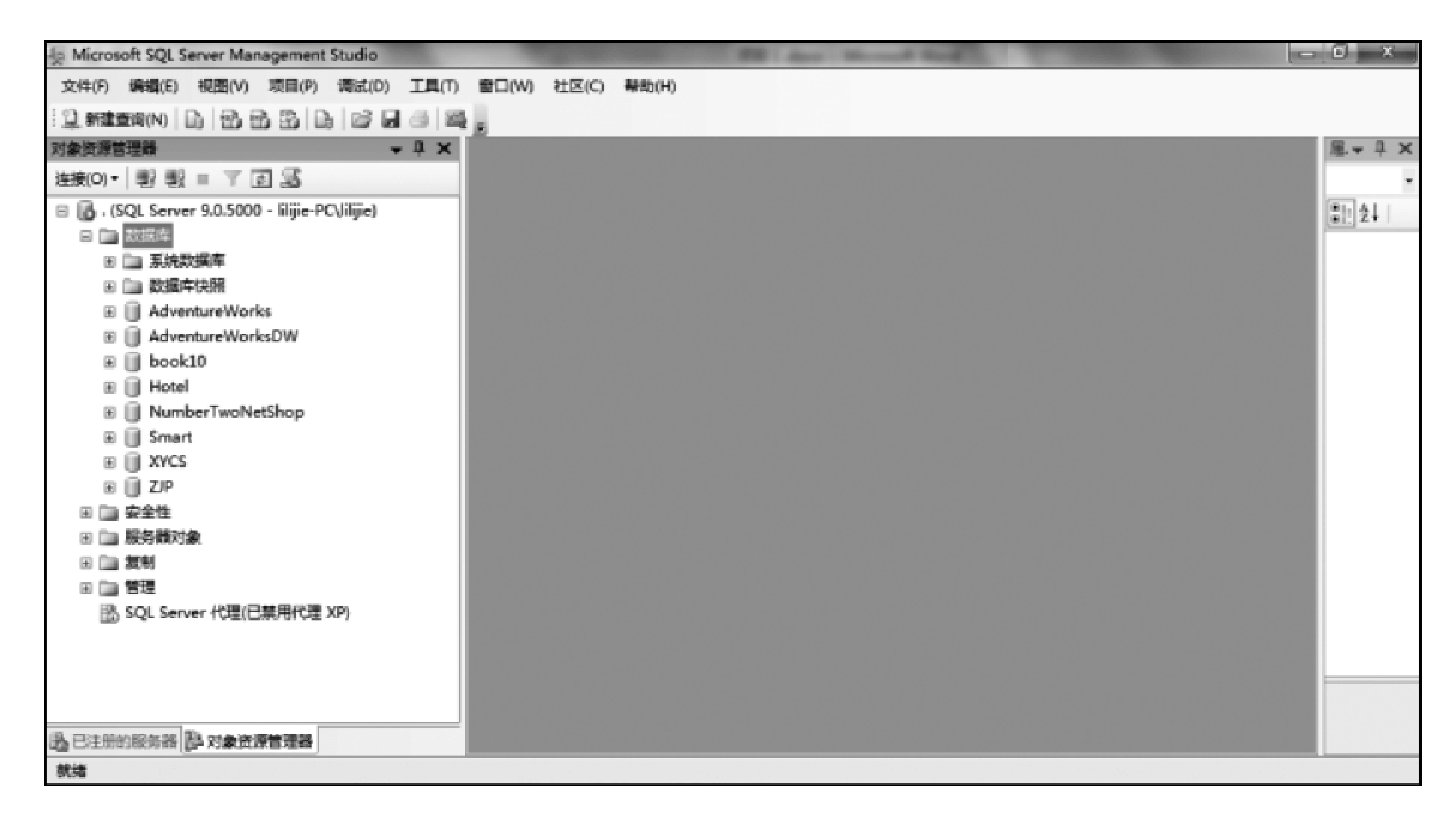


图 1-14 打开“对象资源管理器”窗口

④ 在“对象资源管理器”窗口中，右击“数据库”，在打开的快捷菜单中选择“新建数据库”命令，打开“新建数据库”窗口，如图 1-15 所示。

⑤ 在“新建数据库”窗口的“数据库名称”文本框中输入 database_demo，SQL Server Management Studio 会自动生成其他部分，如数据文件名、日志文件名。单击“确定”按钮，创建一个新的数据库 database_demo。

(2) 使用 Create Database 语句创建数据库。

除了上述图形化方式创建数据库外，还可以采用 Create Database 语句来创建数据库。

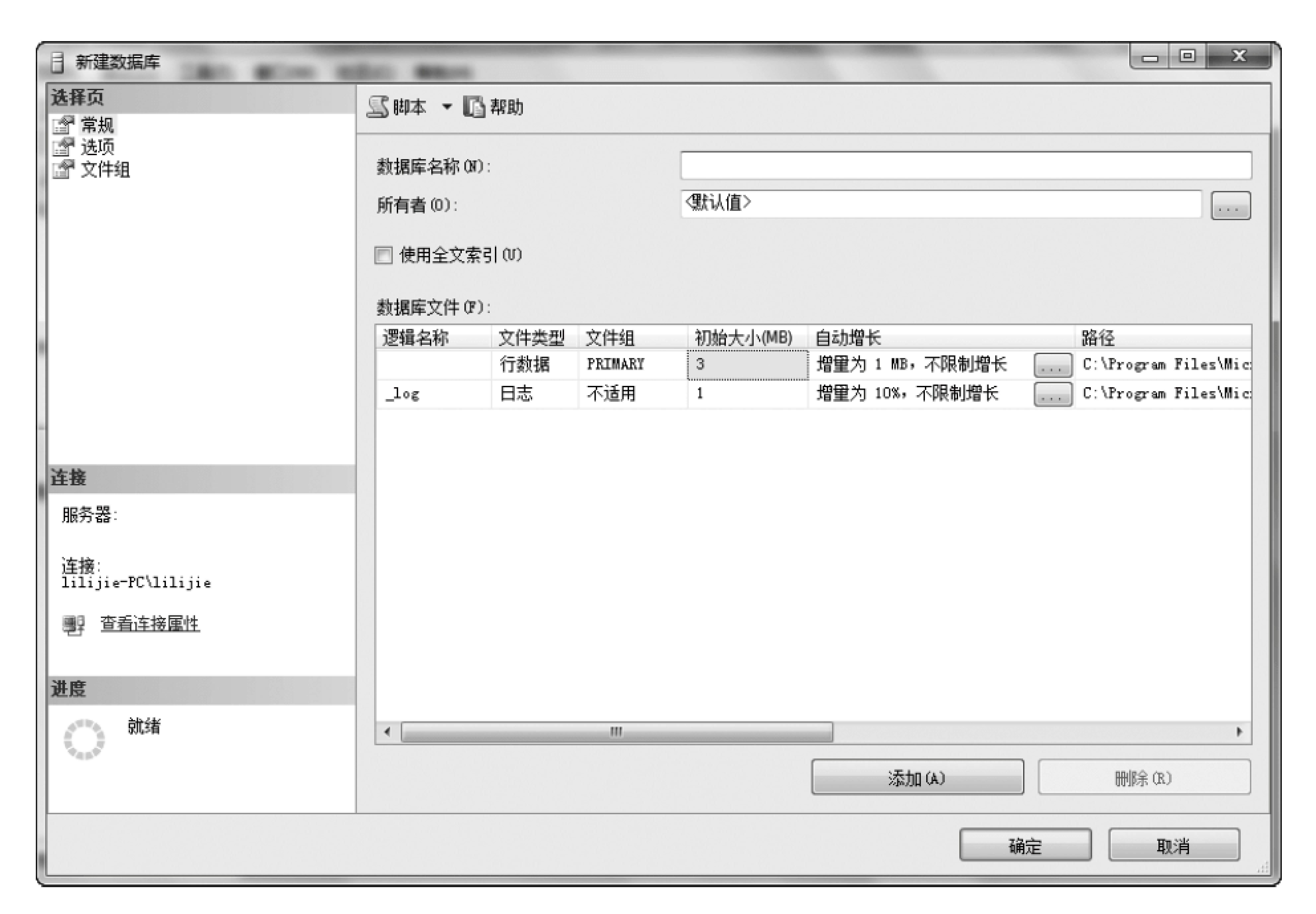


图 1-15 “新建数据库”窗口

采用 Create Database 语句的优势之一是可以保存相应的 T-SQL 语句，以方便批量操作。Create Database 语法如下。

```
CREATE DATABASE database_name
[ON
(
    NAME=logical_file_name,
    FILENAME='os_file_name'
    [,SIZE=size[KB|MB|GB|TB]]
    [,MAXSIZE={max_size[KB|MB|GB|TB]|UNLIMITED}]
    [,FILEGROWTH=growth_increment[KB|MB|GB|TB|%]]
)]
[LOG ON
(
    NAME=logical_file_name,
    FILENAME='os_file_name'
    [,SIZE=size[KB|MB|GB|TB]]
    [,MAXSIZE={max_size[KB|MB|GB|TB]|UNLIMITED}]
    [,FILEGROWTH=growth_increment[KB|MB|GB|TB|%]]
)
];
```

其中[]表示可选项，database_name 指的是新数据库的名称。数据库名称在 SQL Server 的实例中必须唯一，并且必须符合标识符规则。除非没有为日志文件指定逻辑名称，否则 database_name 最多可以包含 128 个字符。如果未指定逻辑日志文件名称，则 SQL Server 将通过向 database_name 追加后缀来为日志生成 logical_file_name 和 os_file_name。这

会将 database_name 限制为 123 个字符,从而使生成的逻辑文件名称不超过 128 个字符。如果未指定数据文件的名称,则 SQL Server 使用 database_name 作为 logical_file_name 和 os_file_name。默认路径从注册表中获得。可以使用 Management Studio 中的【服务器属性】(【数据库设置】页)更改默认路径。更改默认路径要求重新启动 SQL Server。

ON 指定显式定义用来存储数据库数据部分的磁盘文件(数据文件)。当后面是以逗号分隔的用以定义主文件组的数据文件项列表时,需要使用 ON。

LOG ON 指定显式定义用来存储数据库日志的磁盘文件(日志文件)。LOG ON 后跟以逗号分隔的用以定义日志文件的项列表。如果没有指定 LOG ON,将自动创建一个日志文件,其大小为该数据库的所有数据文件大小总和的 25%或 512 KB,取两者之中的较大者。不能对数据库快照指定 LOG ON。

示例 1:创建未指定文件的数据库。

以下示例创建名为 database1 的数据库,并创建相应的主文件和事务日志文件。因为语句没有<filespec>选项,所以主数据库文件的大小为 model 数据库主文件的大小。事务日志将设置为下列值中的较大者:512KB 或主数据文件大小的 25%。因为没有指定 MAXSIZE,文件可以增大到填满所有可用的磁盘空间为止。此示例演示如何在创建 database1 数据库之前删除名为 database1 的数据库(如果它存在)。

```
USE master;
GO
CREATE DATABASE database1;
```

示例 2:创建指定数据和事务日志文件的数据库。

下面的示例将创建数据库 Sales。因为没有使用关键字 PRIMARY,第一个文件(Sales_dat)将成为主文件。因为在 Sales_dat 文件的 SIZE 参数中没有指定 MB 或 KB,将使用 MB 并按 MB 分配。Sales_log 文件以 MB 为单位进行分配,因为 SIZE 参数中显式声明了 MB 后缀。

```
CREATE DATABASE Sales
ON (NAME= Sales_dat,    FILENAME= 'C:\Program Files\Microsoft SQL
    Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\saledat.mdf',    SIZE= 10,    MAXSIZE= 50,    FILEGROWTH
    = 5 )
LOG ON
{NAME= Sales_log,    FILENAME= 'C:\Program Files\Microsoft SQL
    Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\salelog.ldf',    SIZE= 5MB,    MAXSIZE= 25MB,
    FILEGROWTH= 5MB};
```

1.4.6 创建表

在关系型数据库中,表是数据的组织形式,也是最基本的数据库对象。表是由行、列构成的二维结构。在 SQL Server 2008 中,可以通过 SQL Server Management Studio 提供的图形界面工具或 T-SQL 相关语句对表进行创建、修改和删除等。在具体创建表之前,先来

理解基本数据类型。

（1）理解数据类型

为列选择数据类型时，应选择期望存储的所有数据值的数据类型，同时使所需的空间量最小。SQL Server 数据类型有 7 类，如表 1-1 所示。

表 1-1　SQL Server 的 7 类数据类型

数据类型分类	基 本 目 的
精确数字	存储带小数或不带小数的精确数字
近似数字	存储带小数或不带小数的数值
货币	存储带小数位的数值；专门用于货币值，最多可以有 4 个小数位
日期和时间	存储日期和时间信息，并强制实施特殊的年代规则，如拒绝 2 月 30 日这个值
字符	存储基于字符的可变长度的值
二进制	存储以严格的二进制（0 和 1）表示的数据
专用数据类型	要求专门处理的复杂数据类型，诸如 XML 文档或者全局唯一的标识符（GUID）

① 精确数字数据类型。精确数字数据类型用来存储没有小数位或有多个小数位的数值。使用任何算术运算符都可以操纵这些数据类型中存储的数值，而不需要任何特殊处理。精确数字数据类型的存储也是精确定义的，因此，无论是 Intel 处理器架构还是 AMD 处理器架构，这些数据类型中存储的任何数据都返回和计算得到相同的值。表 1-2 列出了 SQL Server 支持的精确数字数据类型。

表 1-2　精确数字数据类型

数据类型	存　储	值　域	作　用
bigint	8 字节	－2E63～2E63－1	存储非常大的正、负整数
int	4 字节	－2E31～2E31－1	存储正、负整数
smallint	2 字节	－32 768～32 767	存储正、负整数
tinyint	1 字节	0～255	存储小范围的正整数
decimal（p，s）	依据不同的精度，需要 5～17 字节	－10E38＋1～10E38－1	最大可以存储 38 位十进制数
numeric（p，s）	依据不同的精度，需要 5～17 字节	－10E38＋1～10E38－1	功能上等价于 decimal，并可以与 decimal 交换使用

decimal 和 numeric 数据类型接受参数来完成数据类型定义。这些参数定义数据类型的精度和小数位数。例如，decimal（12，4）定义了一个总共有 12 位数字的十进制值，其中小数点后面有 4 位数字。

在这组数据类型中，int 和 dedcimal 是最常用的数据类型。使用 decimal 数据类型可以存储整型值，但这么做每行需要额外的存储字节，因此不要这么使用 decimal 数据类型。如果在一个列中打算存储的值的范围不超过 32 767，则通过使用 smallint 代替 int，每行可以节省 2 字节。如果取值范围只是在 0～255，则通过使用 tinyint 数据类型，每行可以节省 3 字节。

② 近似数字数据类型。近似数字数据类型可以存储十进制值。然而，float 或 real 数据类型中存储的数据，只能精确到数据类型定义中指定的精度。不能保证小数点右边的所有数字都被正确存储。例如，如果把 1.000 154 54 存储在一个定义为 float(8)的数据类型中，则该列只能保证精确地返回 1.000 154。SQL Server 存储数据时对小数点右边的数进行四舍五入。因此，涉及这些数据类型的计算，会出现舍入误差。在 Intel 处理器和 AMD 处理器之间传输包含涉及这些数据类型的表的数据库时，也会引入误差。表 1-3 列出了 SQL Server 支持的近似数字数据类型。

表 1-3　近似数字数据类型

数据类型	存储	取值范围	作　用
float(p)	4 或 8 字节	−2.23E308～2.23E308	存储大型浮点数，超过十进制数据类型的容量
real	4 字节	−3.4E38～3.4E38	仍然有效，但为了满足 SQL-92 标准，已经被 float 替换了

float 数据类型在定义时接受一个参数，该参数决定了精确存储的位数。例如，一个 float(8)列精确存储 7 位数字，任何超过该数的位数都会遭遇舍入误差。由于这些数据类型是不精确的，所以几乎不使用它们。只有在精确数据类型不够大且不能存储数值时，才可以考虑使用 float。

③ 货币数据类型。货币数据类型旨在存储精确到 4 个小数位的货币值。表 1-4 列出了 SQL Server 支持的货币数据类型。

表 1-4　货币数据类型

数据类型	存储空间	取 值 范 围	作　用
money	8 字节	−922 337 203 685 477.580 8～922 337 203 685 477.580 7	存储大型货币值
smallmoney	4 字节	−214 748.364 8～214 748.364 7	存储小型货币值

在数据库中几乎不定义 smallmoney 数据类型，尽管对很多处理产品和订单的应用程序而言这种数据类型是最精确的选择。由于不正确地使用了 money 数据类型，使每行数据浪费了 4 字节的存储空间，这种情况是比较普遍的。虽然 money 和 smallmoney 数据类型旨在存储货币值，但在金融应用程序中几乎不使用它们。相反，这些应用程序使用 decimal 数据类型，因为它们需要执行精确到 6 个、8 个甚至 12 个小数位的计算。

④ 日期和时间数据类型。表 1-5 列出了 SQL Server 支持的日期和时间数据类型。

表 1-5　日期和时间数据类型

日期类型	存储空间	取 值 范 围	作　用
datetime	8 字节	从“January 1，1753”到“December 31，9999”，精度为 3.33ms	存储大型日期和时间值
smalldatetime	4 字节	从“January 1，1900”到“June 6，2079”，精度为 1min	存储较小范围的日期和时间值

datetime 和 smalldatetime 数据类型在计算机内部是作为整数存储的。datetime 数据类型存储为一对 4 字节整数，它们一起表示自 1753 年 1 月 1 日午夜 12 点钟经过的毫秒数。前 4 字节存储日期，而后 4 字节存储时间。smalldatetime 数据类型存储为一对 2 字节整数，

它们一起表示自 1900 年 1 月 1 日午夜 12 点钟经过的分钟数。前 2 字节存储日期,后两个日期存储时间。

⑤ 字符数据类型。存储字符数据时,选择一种为此目的而设计的数据类型。每种字符数据类型使用 1 个或 2 字节存储每个字符,具体取决于该数据类型使用 ANSI(American National Standards Institute)编码还是 Unicode 编码。

Unicode 数据类型前有一个 n。例如,nchar 是 Unicode 数据类型,对应于使用 ANSI 编码的 char 数据类型。定义一个字符数据类型时,指定该列允许存储的最大字节数。例如,char(10)最多可以存储 10 个字符,因为每个字符要求 1 字节的存储空间,而 nchar(10)最多可以存储 5 个字符,因为每个 Unicode 字符要求使用 2 字节的存储空间。表 1-6 列出了 SQL Server 支持的字符数据类型。

表 1-6 字符数据类型

数据类型	存储空间	字 符 数	作 用
char(n)	1～8000 字节	最多 8000 个字符	固定宽度的 ANSI 数据类型
nchar(n)	2～8000 字节	最多 4000 个字符	固定宽度的 Unicode 数据类型
varchar(n)	1～8000 字节	最多 8000 个字符	固定宽度的 ANSI 数据类型
varchar(max)	最大 2GB	最多 1 073 741 824 个字符	可变宽度的 ANSI 数据类型
nvarchar(n)	2～8000 字节	最多 4000 个字符	可变宽度的 Unicode 数据类型
nvarchar(max)	最大 2GB	最多 536 870 912 个字符	可变宽度的 Unicode 数据类型
text	最大 2GB	最多 1 073 741 824 个字符	可变宽度的 ANSI 数据类型
ntext	最大 2GB	最多 536 870 912 个字符	可变宽度的 Unicode 数据类型

为什么有那么多看起来好像相互等价的字符数据类型呢?数据类型的区别可能不怎么明显,但是它们是重要的。一个 char 数据类型,无论是 ANSI 标准还是 Unicode 标准,都是固定宽度的数据类型。因此,不管列中存储多少个字符,它们总是消耗相同的存储空间。例如,一个 char(30)列使用 30 字节存储空间,而不管在该列存储 1 个字符还是 30 个字符。任何未被使用的空间都用空格填补,直到填满为该列指定的存储空间。然而,一个 varchar(30)列对该列中存储的每个字符只用 1 字节。

text 和 ntext 数据类型旨在存储大量基于字符的数据。然而,text 和 ntext 列允许的操作不是很多。例如,不能使用等于运算符比较它们,也不能连接它们。很多系统函数也不能使用 text 和 ntext 数据类型。

⑥ 二进制数据类型。有很多时候需要存储二进制数据。因此,SQL Server 提供了三种二进制数据类型,允许在一个表中存储各种数量的二进制数据。表 1-7 列出了 SQL Server 支持二进制数据类型。

表 1-7 二进制数据类型

数据类型	存储空间	作 用
binary(n)	1～8000 字节	存储固定大小的二进制数据
varbinary(n)	1～8000 字节	存储可变大小的二进制数据
varbinary(max)	最多 2GB	存储可变大小的二进制数据
image	最多 2GB	存储可变大小的二进制数据

二进制数据类型基本上用来存储 SQL Server 中的文件。binary/varbinary 数据类型用来存储小文件,诸如一组 4KB 或 6KB 文件,其中包含各种以本机格式表示的数据文件。

image 数据类型是这组数据类型中最流行的数据类型。虽然可以用 image 数据类型存储图片,但也可以使用这种数据类型存储 Word、Excel、PDF 和 Visio 文档。使用 image 数据类型可以存储任何一个小于或等于 2GB 的文件。这种数据类型的最著名的实现之一是 TerraServer 项目,这是一个高达几 TB 的陆地图形数据库,可以在 www. terraserver. com 上访问它。varbinary(max)数据类型是 SQL Server 2005 新增的一种数据类型。它可以存储与 image 数据类型相同大小的数据,并且可以使用它执行所有可以用 binary/varbinary 数据类型执行的操作和函数。

⑦ 特殊数据类型。除了上述标准数据类型外,SQL Server 还提供了另外 7 种特殊数据类型。表 1-8 描述了这些特殊数据类型。

表 1-8　特殊数据类型

数据类型	作　　用
bit	存储 0、1 或 null。用于基本“标记”值。TRUE 被转换为 1,而 FALSE 被转换为 0
timestamp	一个自动生成的值。每个数据库都包含一个内部计数器,指定一个不与实际时钟关联的相对时间计数器。一个表只能有一个 timestamp 列,并在插入或修改行时被设置为数据库时间戳
uniqueidentifier	一个 16 位 GUID,用来全局标识数据库、实例和服务器中的一行
sql_variant	可以根据其中存储的数据改变数据类型。最多存储 8000 字节
cursor	供声明游标的应用程序使用。它包含一个可用于操作的游标的引用。该数据类型不能在表中使用
table	用来存储随后进行的处理的结果集。该数据类型不能用于列。该数据类型的唯一使用时机是在触发器、存储过程和函数中声明表变量时
Xml	存储一个 XML 文档,最大大小为 2GB。可以指定选项,强制只能存储格式良好的文档

(2) 使用 SQL Server Management Studio 提供的图形界面工具创建表

SQL Server 2008 提供两种创建表的途径。其中一种是使用 SQL Server Management Studio 提供的图形界面工具创建表。

示例 3:利用 SQL Server Management Studio 图形界面工具,在名为 database1 的数据库下建一个名字为 Customer 的表。该示例可以按以下步骤完成。

① 打开 SQL Server Management Studio 图形界面工具,输入用户名和密码,连接数据库实例。

② 在“对象资源管理器”窗口中双击“数据库”,右击 database1 节点下的“表”,选择“新建表”命令,如图 1-16 所示。

③ 接着在右边的表设计器中输入列名,指定相应的数据类型以及是否允许空值,如图 1-17 所示。

④ 右击第一行中的 Customer_Id,从快捷菜单中选择“设置主键”命令,将 Customer_Id 设置为主键。

⑤ 单击工具栏上的“保存”按钮,打开“保存”对话框,输入表名 Customer,单击“确定”按钮,则表创建成功。

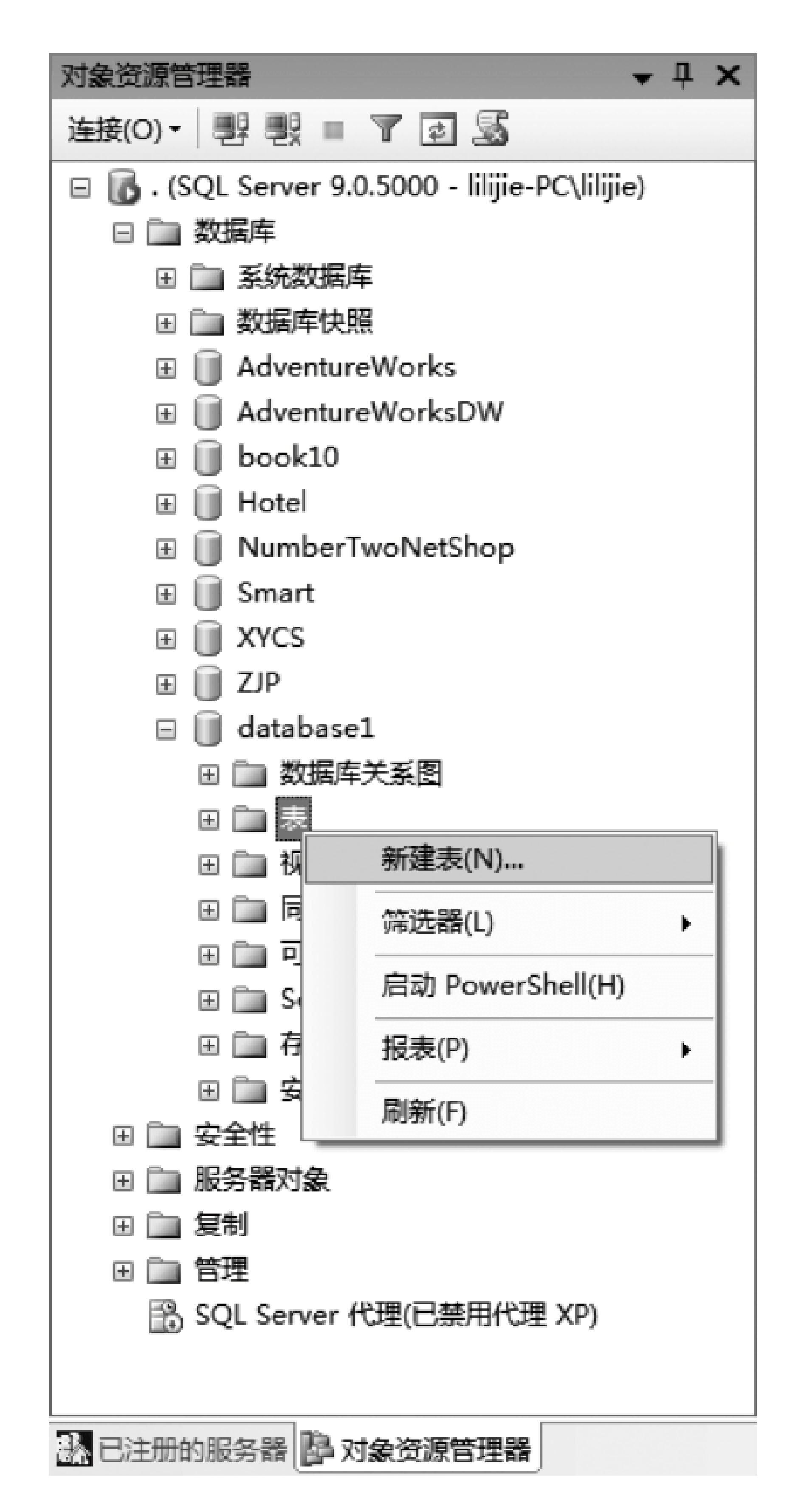


图 1-16　“新建表”命令

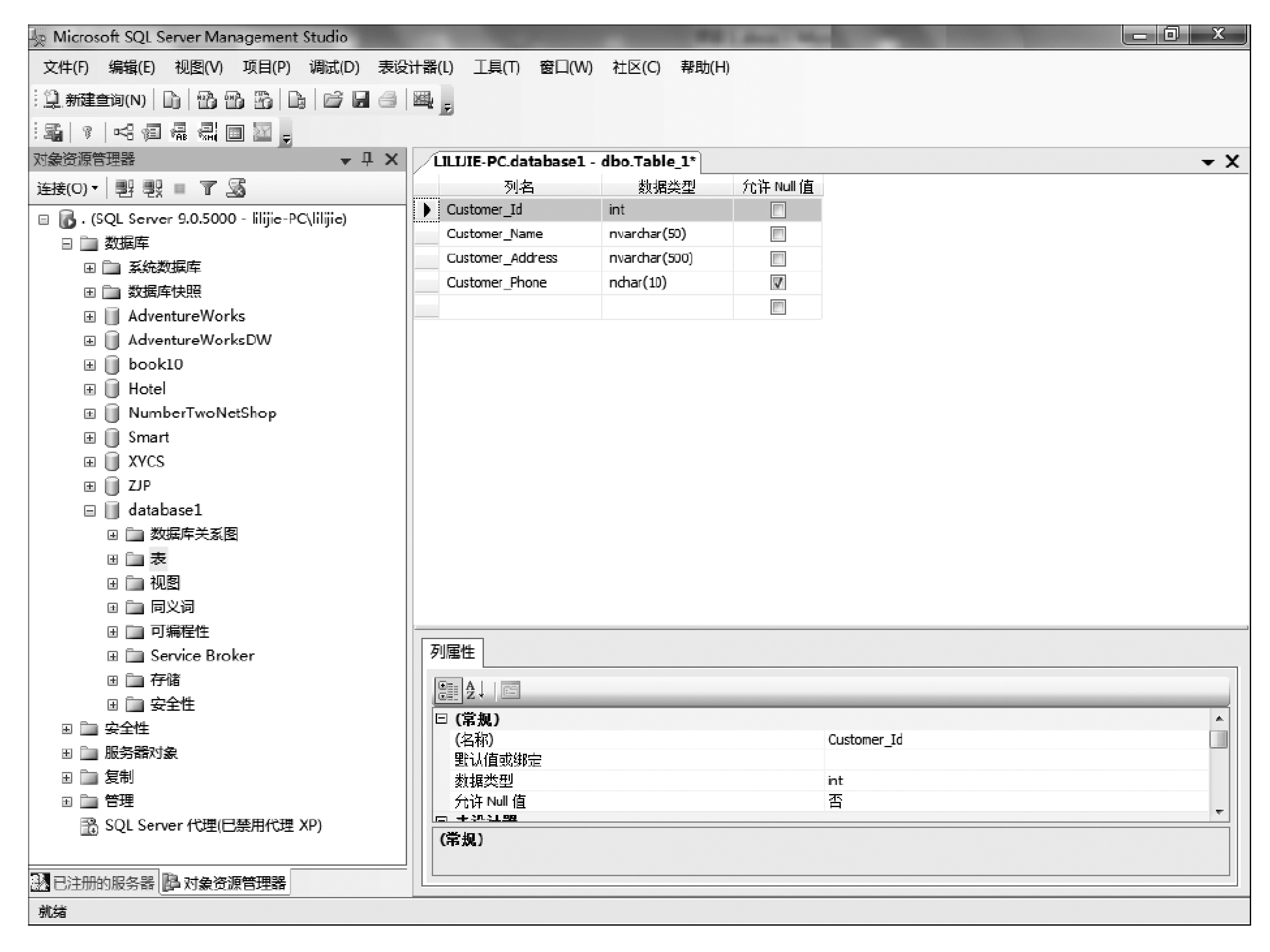


图 1-17　表设计器

(3) 使用 CREATE TABLE 语句创建表

CREATE TABLE 语句用于创建数据库中的表。其格式如下：

```
CREATE TABLE 表名称
(
    列名称 1 数据类型,
    列名称 2 数据类型,
    列名称 3 数据类型,
    ⋮
)
```

示例 4：创建名字为 Person 的表，该表包含 5 个列，列名分别是：Id_P、LastName、FirstName、Address 以及 City。

```
CREATE TABLE Persons
(
    Id_P int primary key,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
)
```

1.4.7 创建外键约束

外键(FOREIGN KEY,K)是用于建立和加强两个表数据之间的链接的一列或多列。当创建或修改表时，可通过定义 FOREIGN KEY 约束来创建外键。在外键引用中，当一个表的列被引用并作为另一个表的主键值的列时，就在两表之间创建了链接。这个列就成为第二个表的外键。如图 1-18 所示，因为销售订单和销售人员之间存在一种逻辑关系，所以 AdventureWorks 数据库中的 Sales. SalesOrderHeader 表含有一个指向 Sales. SalesPerson 表的链接。SalesOrderHeader 表中的 SalesPersonID 列与 SalesPerson 表中的主键列相对应。SalesOrderHeader 表中的 SalesPersonID 列是指向 SalesPerson 表的外键。

FOREIGN KEY 约束并不仅仅可以与另一表的 PRIMARY KEY 约束相链接，它还可以定义为引用另一表的 UNIQUE 约束。FOREIGN KEY 约束可以包含空值，但是，如果任何组合 FOREIGN KEY 约束的列包含空值，则将跳过组成 FOREIGN KEY 约束的所有值的验证。若要确保验证了组合 FOREIGN KEY 约束的所有值，可将所有参与列指定为 NOT NULL。FOREIGN KEY 约束的主要目的是控制可以存储在外键表中的数据，但它还可以控制对主键表中数据的更改。例如，如果在 Sales. SalesPerson 表中删除一个销售人员所在的行，而这个销售人员的 ID 由 Sales. SalesOrderHeader 表中的销售订单使用，则这两个表之间关联的完整性将被破坏；SalesOrderHeader 表中删除的销售人员的销售订单因为与 SalesPerson 表中的数据没有链接而变得孤立了。FOREIGN KEY 约束有效防止这种情况的发生。如果主键表中数据的更改使之与外键表中数据的链接失效，则这种更改将无法实现，从而确保了引用的完整性。如果试图删除主键表中的行或更改主键值，而该主键值与另一表的 FOREIGN KEY 约束中的值相对应，则该操作将失败。若要成功更改或删除 FOREIGN KEY 约束的行，必须先在外键表中删除或更改外键数据。SQL Server 对一个

图 1-18　Sales. SalesOrderHeader 表中的外键 SalesPerson

表可以包含的 FOREIGN KEY 约束(引用其他表)数没有预定义限制,对引用特定表的其他表所拥有的 FOREIGN KEY 约束数也没有预定义的限制。但是,实际的 FOREIGN KEY 约束数会受到硬件配置以及数据库和应用程序的设计的限制。建议表中包含的 FOREIGN KEY 约束不要超过 253 个,并且引用该表的 FOREIGN KEY 约束也不要超过 253 个。添加外键约束语法如下:

```
ALTER TABLE 表名 ADD FOREIGN KEY(列名)REFERENCES 主表名(主表中的主键)
```

示例 5:根据图 1-19 所示,为 Orders 表的 Id_P 列创建外键约束。

"Persons" 表:

Id_P	LastName	FirstName	Address	City
1	Adams	John	Oxford Street	London
2	Bush	George	Fifth Avenue	New York
3	Carter	Thomas	Changan Street	Beijing

"Orders" 表:

Id_O	OrderNo	Id_P
1	77 895	3
2	44 678	3
3	22 456	1
4	24 562	1

图 1-19　Orders 表外键

```
Alter Table Orders add foreign key(Id_P)references Persons(Id_P)
```

1.4.8 创建唯一性约束

唯一性(UNIQUE)约束唯一标识数据库表中的每条记录。UNIQUE 和 PRIMARY KEY 约束均为列或列集合提供了唯一性的保证。PRIMARY KEY 拥有自动定义的 UNIQUE 约束。请注意,每个表可以有多个 UNIQUE 约束,但是每个表只能有一个 PRIMARY KEY 约束。其语法如下:

```
ALTER TABLE 表名 ADD UNIQUE(列名)
```

示例 6:为 Orders 表的 OrderNo 列创建唯一性约束,确保 OrderNo 列数据不重复。

```
Alter Table Orders add unique(OrderNo)
```

1.5 项目实施

1.5.1 任务 1:创建 Smart On Line 电子商城数据库

项目小组经过小组讨论,最终形成 Smart On Line 电子商城数据库说明文档。项目负责人现在要求数据库开发人员根据说明文档创建相应数据库。数据库各表具体说明见表 1-9～表 1-17。

表 1-9　T_Customer 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	Customer_ID	int	4	0	是	是		否		
2	Customer_Account	nvarchar	50	0				否		
3	Customer_Pwd	nvarchar	50	0				否		
4	Customer_Age	int	4	0				否		
5	Customer_Phone	nvarchar	50	0				是	是	
6	Customer_Email	nvarchar	50	0				是	是	

表 1-10　T_CustomerAddress 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	CustomerAddress_ID	int	4	0	是	是		否		
2	Customer_ID	int	4	0			是	否		
3	Customer_Address	nvarchar	50	0				否		

表 1-11　T_DetailsType 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	DTID	int	4	0	是	是		否		
2	DTName	nvarchar	50	0				否		
3	DTTableName	nvarchar	50	0				否		
4	DTBeizu	nvarchar	50	0				否		

表 1-12　T_Level1 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	Level1_ID	int	4	0	是	是		否		
2	Level1_Name	nvarchar	50	0				否	是	
3	Level1_Desc	nvarchar	500	0				是		

表 1-13　T_Level2 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	Level2_ID	int	4	0	是	是		否		
2	Level2_Name	nvarchar	50	0				否	是	
3	Level2_Desc	nvarchar	500	0				是		
4	Level1_ID	int	4	0			是	是		

表 1-14　T_Level3 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	Level3_ID	int	4	0	是	是		否		
2	Level3_Name	nvarchar	50	0				否	是	
3	Level3_Desc	nvarchar	500	0				是		
4	Level2_ID	int	4	0			是	是		
5	Level1_ID	int	4	0			是	是		

表 1-15　T_Order 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	Order_ID	int	4	0	是	是		否		
2	Order_Number	nchar	10	0				否	是	
3	CustomerAddress_ID	int	4	0			是	否		
4	Pay_Form	nchar	10	0				否		
5	Is_Send	bit	1	0				是		
6	Is_Pay	bit	1	0				否		
7	Pay_Time	datetime	8	3				否		

表 1-16　T_ShoppingCart 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	ShoppingItem_ID	int	4	0	是	是		否		
2	Customer_ID	int	4	0			是	否		
3	Ware_ID	int	4	0			是	否		
4	Ware_Qty	int	4	0				否		
5	Ware_Sum	float	8	0				是		
6	Order_ID	int	4	0			是	是		

表 1-17　T_Ware 表

序号	列　名	数据类型	长度	小数位	标识	主键	外键	允许空	唯一性	说明
1	Ware_ID	int	4	0	是	是		否		
2	Ware_Number	nchar	10	0				否	是	
3	Ware_Name	nvarchar	200	0				否		
4	Ware_Weight	nchar	10	0				否		
5	Ware_Stock	int	4	0				是		
6	Ware_Level3	int	4	0				否		
7	Ware_Price	float	8	0				是		
8	Extend_ID	int	4	0			是	是		
9	Ware_Image	nvarchar	50	0				是		
10	DetailsType_ID	int	4	0			是			

1. 任务目标

（1）能利用 SQL Server 2008 创建数据库。

（2）能根据数据库说明文档创建表。

2. 任务内容

（1）创建 Smart 数据库。

（2）在 Smart 数据库中创建表。

3. 任务实施步骤

（1）打开 SQL Server Management Studio 工具，单击工具栏中的“新建查询”按钮，打开查询分析器窗口，如图 1-20 所示。

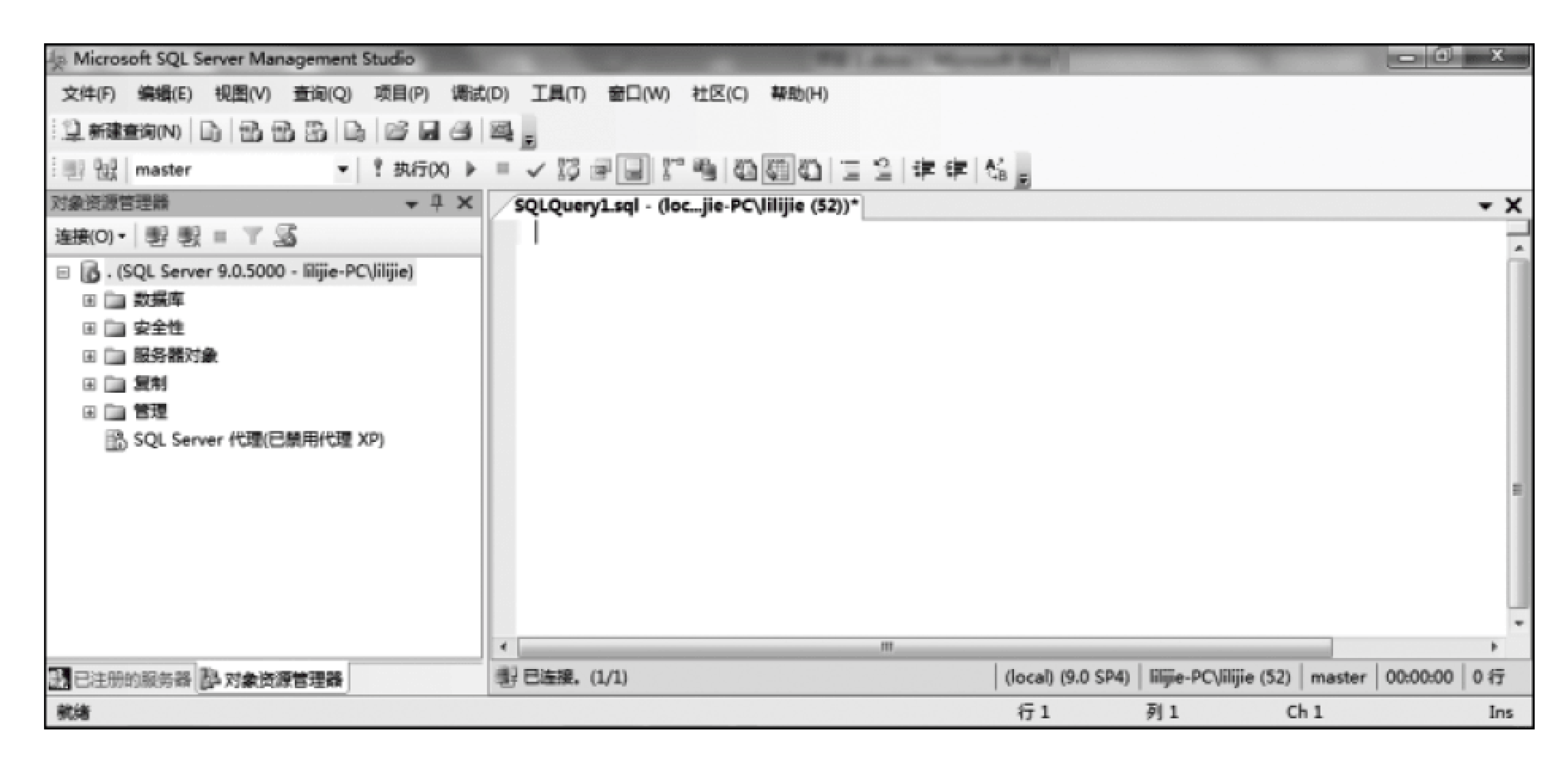


图 1-20　查询分析器窗口

（2）在查询分析器窗口中输入 Create Database Smart，单击工具栏中的“执行”按钮，创建 Smart 数据库。

（3）在查询分析器窗口中输入 Use Smart，单击工具栏中的“执行”按钮或者按 F5 键，将当前活动数据库切换至 Smart 数据库。

（4）接下来创建简单表。继续在查询分析器中输入下列创建表的 T-SQL 代码，输入完毕后单击工具栏中的“执行”按钮。

```
CREATE TABLE T_Customer(
      Customer_ID int IDENTITY(1,1) PRIMARY KEY,
      Customer_Account nvarchar(50) NOT NULL,
      Customer_Pwd nvarchar(50) NOT NULL,
      Customer_Age int NOT NULL,
      Customer_Phone nvarchar(50) NULL,
      Customer_Email nvarchar(50) NULL
)
CREATE TABLE T_CustomerAddress(
      CustomerAddress_ID int IDENTITY(1,1) PRIMARY KEY ,
      Customer_ID int NOT NULL,
      Customer_Address nvarchar(50) NOT NULL
)
CREATE TABLE T_DetailsType(
      DTID int IDENTITY(1,1)PRIMARY KEY,
      DTName nvarchar(50)NOT NULL,
      DTTableName nvarchar(50)NOT NULL,
      DTBeizu nvarchar(50)NOT NULL,
)
CREATE TABLE T_Level1(
      Level1_ID int IDENTITY(1,1)PRIMARY KEY,
      Level1_Name nvarchar(50)NOT NULL,
      Level1_Desc nvarchar(500)NULL
)

CREATE TABLE T_Level2(
      Level2_ID int IDENTITY(1,1)PRIMARY KEY,
      Level2_Name nvarchar(50)NOT NULL,
      Level2_Desc nvarchar(500)NULL,
      Level1_ID int NULL
)
CREATE TABLE T_Level3(
      Level3_ID int IDENTITY(1,1)PRIMARY KEY,
      Level3_Name nvarchar(50)NOT NULL,
      Level3_Desc nvarchar(500)NULL,
      Level2_ID int NULL,
      Level1_ID int NULL
)
CREATE TABLE T_Order(
      Order_ID int IDENTITY(1,1)PRIMARY KEY,
      Order_Number nchar(10)NOT NULL,
      CustomerAddress_ID int NOT NULL,
      Pay_form nchar(10)NOT NULL,
      Is_Send bit NULL,
      Is_Pay bit NOT NULL,
      Pay_Time datetime NOT NULL
)
CREATE TABLE T_ShoppingCart(
      ShoppingItem_ID int IDENTITY(1,1)PRIMARY KEY,
      Customer_ID int NOT NULL,
      Ware_ID int NOT NULL,
```

```
        Ware_Qty int NOT NULL,
        Ware_Sum float NULL,
        Order_ID int NULL
)
CREATE TABLE T_Ware(
        Ware_ID int IDENTITY(1,1)PRIMARY KEY,
        Ware_Number nchar(10)NOT NULL,
        Ware_Name nvarchar(200)NOT NULL,
        Ware_Weight nchar(10)NOT NULL,
        Ware_stock int NULL,
        Ware_Level3 int NOT NULL,
        Ware_Price float NULL,
        Extend_ID int NULL,
        Ware_Image nvarchar(50)NULL,
        DetailsType_ID int NOT NULL
)
```

1.5.2 任务2：创建表外键约束

1. 任务目标

（1）能使用 Alter Table 语句创建表外键约束。

（2）能使用 SQL Server Management Studio 工具可视化方式创建表外键约束。

2. 任务内容

根据任务 1 中给出的数据库说明书创建表外键约束。

3. 任务实施步骤

创建表外键约束可以采用 ALTER TABLE 语句或者 SQL Server Management Studio 工具的可视化界面两种不同方式。在任务 1.5.2 实施过程中，将介绍采用 SQL Server Management Studio 工具以可视化方式创建 T_CustomerAddress 表的 Customer_ID 列的外键约束。剩余的外键约束将采用 Alter Table 语句实现。

（1）在“对象资源管理器”窗口中，单击“数据库”→Smart→“表”，再右击 T_CustomerAddress，然后选择“设计”命令，打开表设计器窗口，如图 1-21 所示。

（2）右击 Customer_ID 列名，再单击“关系”命令，打开“外键关系”对话框。

（3）单击“添加”按钮，再单击“表和列规范”节点（见图 1-22），打开“表和列”对话框。

（4）单击“主键表”下拉列表，选择 T_Customer 表。接着单击“字段选择”下拉列表，选择 Customer_ID 字段。最后单击“确定”按钮，则 T_CustomerAddress 表的 Customer_ID 列的外键创建完毕。

（5）在“查询分析器”窗口中输入下列代码，完成其他表的外键约束的创建。注意：T_Ware表的 Extend_ID 列参考的主表是后续开发中动态创建的，这个环节暂不需创建外键。

```
alter table T_Level2 add foreign key (Level2_ID)references T_Level1(Level1_ID)
alter table T_Level3 add foreign key (Level2_ID)references T_Level2(Level2_ID)
alter table T_Level3 add foreign key (Level1_ID)references T_Level1(Level1_ID)
alter table T_Order add foreign key (CustomerAddress_ID)
        references T_CustomerAddress(CustomerAddress_ID)
```

```
alter table T_ShoppingCart add foreign key (Customer_Id)
          references T_Customer(Customer_ID)
alter table T_ShoppingCart add foreign key (Order_ID) references T_Order(Order_ID)
alter table T_Ware add foreign key (DetailsType_ID)
          references T_DetailsType(DetailsType_ID)
```

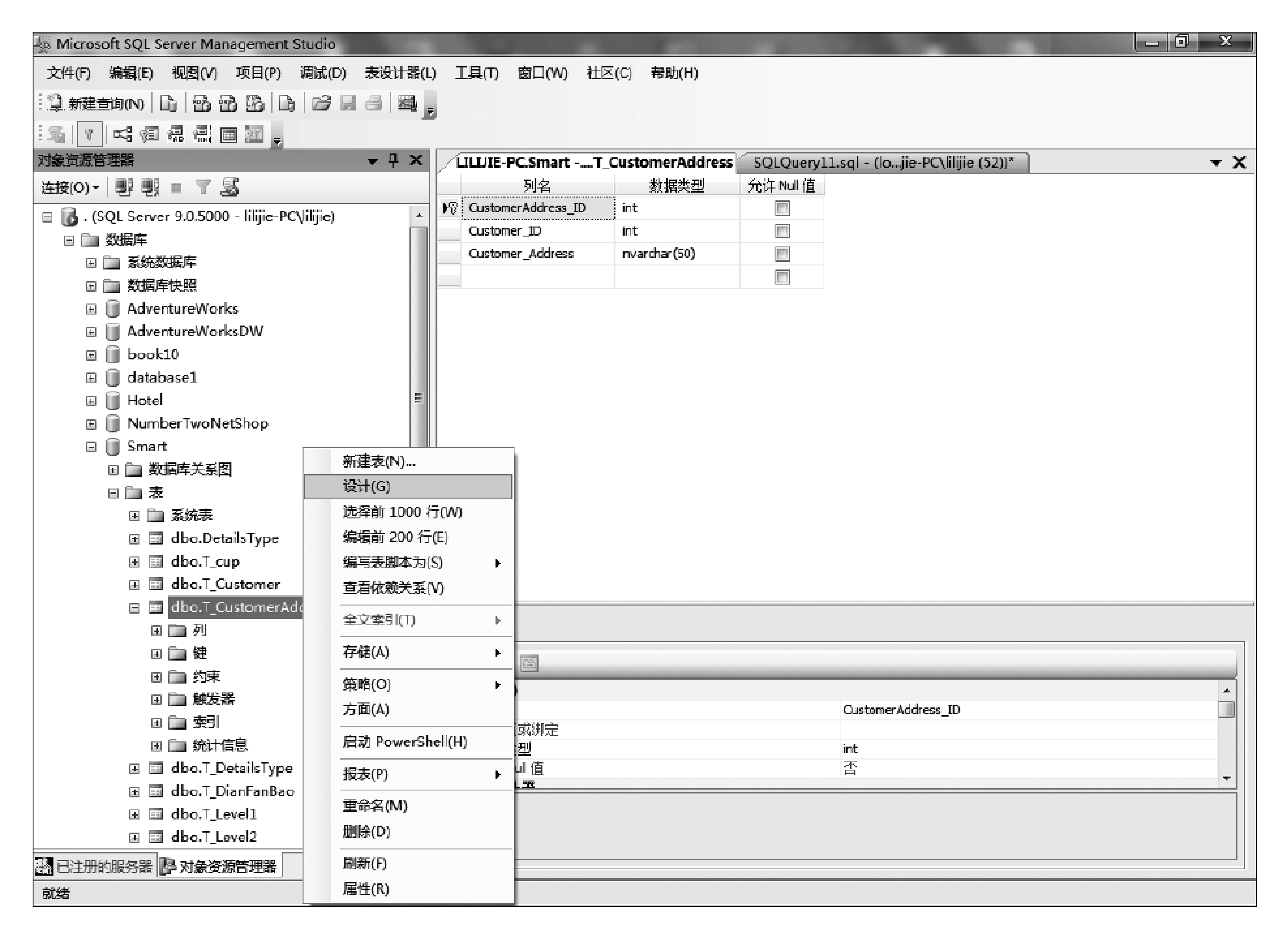


图 1-21　表设计器窗口

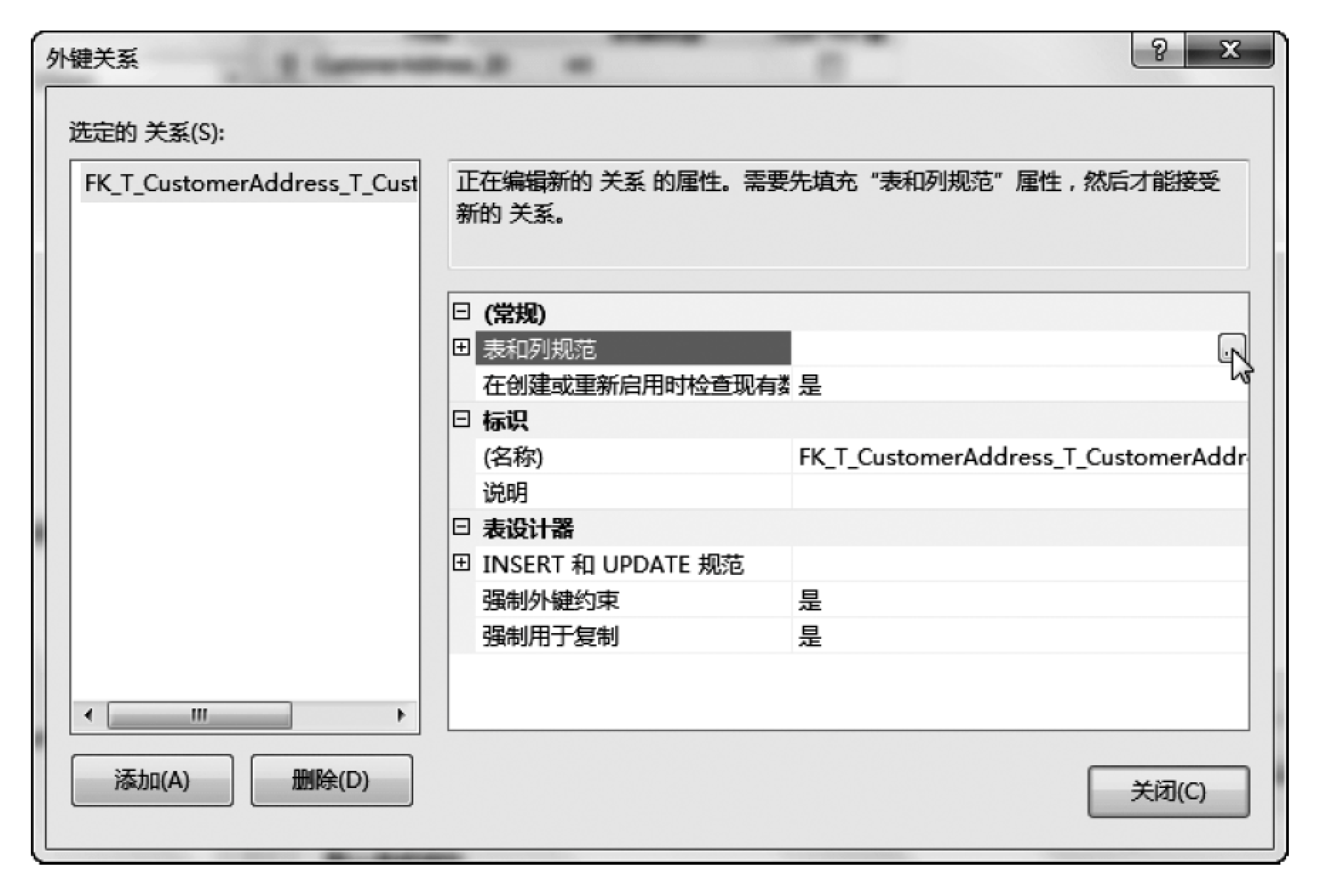


图 1-22　“外键关系”对话框

1.5.3 任务 3：创建唯一性约束

1. 任务目标

能运用 ALTER TABLE 语句创建唯一性约束。

2. 任务内容

根据 1.5.1 中任务 1 给出的数据库说明书，给相关列添加唯一性约束，确保列数据不重复。

3. 任务实施步骤

（1）打开 SQL Server Management Studio 工具，单击工具栏中的“新建查询”按钮，打开“查询分析器”窗口。

（2）在“查询分析器”窗口中输入下列代码并按 F5 键执行命令，确保 T_Customer 表的 Customer_Phone 列和 Customer_Email 列、T_Level1 表的 Level1_Name 列、T_Level2 表的 Level2_Name 列、T_Level3 表的 Level3_Name 列、T_Order 表的 Order_Number 以及 T_Ware 表的 Ware_Number 列的数据不重复。

```
alter table T_Customer add unique(Customer_Phone)
alter table T_Customer add unique(Customer_Email)
alter table T_Level1 add unique(Level1_Name)
alter table T_Level2 add unique(Level2_Name)
alter table T_Level3 add unique(Level3_Name)
alter table T_Order add unique(Order_Number)
alter table T_Ware add unique(Ware_Number)
```

1.6 总 结 归 纳

项目 1 主要介绍了 Smart On Line 电子商城的功能和数据库设计相关的基础知识，包括创建数据库、创建表、创建外键约束和唯一性约束等。通过任务 1、任务 2 和任务 3 的实施，培养学生能自主根据数据库设计说明书创建符合要求的数据库及其相关对象。因篇幅关系，项目 1 中未详细介绍数据库中数据增、删、改操作的语句和相关操作步骤，请大家利用业余时间复习相关的知识。

1.7 课 后 习 题

单项选择题

1. 下列关于用 CREATE TABLE 语句创建数据表的叙述，正确的是（　　）。
   A. 必须在数据表名称中指定表所属的数据库
   B. 必须指明数据表的所有者
   C. 指定的所有者和表名称在数据库中必须唯一
   D. 省略表名称时，自动创建一个临时表

2. 表设计器的“允许空”，用于创建字段的(　　)约束。
A. 主键　B. 外键　C. CHECK　D. 非空
3. 下列字段定义错误的是(　　)。
A. 学号 varchar(16)　B. 人数 int 4
C. 产量 float　D. 价格 decimal(8,2)
4. 不属于 SQL Server 的数据类型是(　　)。
A. 整型数据类型　B. 浮点数据类型
C. 通用型数据类型　D. 字符数据类型
5. 不属于整型数据类型的是(　　)。
A. int　B. smallint　C. tinyint　D. integer
6. 如果数据表中某个字段只包含 1～200 的整数，则该字段最好定义为(　　)。
A. int　B. smallint　C. tinyint　D. bit
7. 如果数据表中某个字段的数据精度要求 8～12 位，则该字段最好定义为(　　)。
A. real　B. smallint　C. float　D. money
8. 某个字段的数据类型定义为 decimal(12,5)，则该字段有(　　)位整数。
A. 12　B. 5　C. 6　D. 7
9. 存储诸如“邮政编码”的字段类型，最好定义为(　　)。
A. Char(6)　B. varChar(6)　C. NChar(6)　D. NvarChar(6)
10. 存储诸如“通信地址”的字段类型，最好定义为(　　)。
A. Binary(n)　B. varChar(n)　C. Nbianry(n)　D. Ntext

1.8 同步操练

公司业务部门接到开发酒店管理系统的业务，其功能性需求描述如表 1-18 所示。项目经理现将酒店管理系统数据库设计说明书通过邮箱转发给数据库开发人员甲，要求其根据数据库设计说明书创建数据库及各表和对象(见酒店管理系统数据库文档)。

表 1-18　酒店管理系统功能性需求

功能执行者	功能名称	描　述
系统管理员	登录	订房顾客、系统管理员、客房前台在输入用户名和密码之后通过系统验证进入相应页面
	订房功能	处理上门订房。订房信息存入系统，并可显示客房使用情况
	查询订房内容	订房顾客在外网登录后，可以查询订房顾客、房间、起始时间、结束时间、定金、是否结账等信息，也可以通过客房部前台查询
	修改订房内容	订房顾客通过客房部前台修改订房顾客信息、房间、起始时间、结束时间、定金、是否结账等信息
	取消订房	订房顾客通过客房部前台取消订房，客房部前台删除此顾客的订房信息
	住宿结算	住宿后，系统通过已记录的内容动态生成账单

续表

功能执行者	功能名称	描　　述
系统管理员	添加系统用户	管理员登录后进入系统管理页面，通过此页面管理员可以添加系统用户客房部前台用户
	查询和修改用户	在此页面，管理员可以查询系统用户的信息和修改系统用户的密码
	删除系统用户	在此页面，管理员可以删除相应用户
	添加客房	客房部用户进入房间添加界面，在输入房间编号、房间类型和房间单价之后可以添加房间
	删除客房	客房部用户进入房间删除界面，页面会列出房间列表，用户通过选中房间，单击“删除”按钮的操作删除房间
用户	修改密码	如果用户觉得之前注册的密码不容易记或太简单，可以修改密码
	修改个人信息	如果用户觉得之前填写的报名信息不准确或不完善，可以修改个人信息
	预订客房	用户可在预订客房页面提前预订自己需要的客房

表 1-19～表 1-32 是酒店管理系统数据库文档。

表 1-19　DengJi

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	logintime	nvarchar	60	0				否		
3	logouttime	nvarchar	60	0				是		
4	kfid	int	4	0				是		
5	fee	decimal	9	0				是	0.0	
6	days	nvarchar	60	0				是		

表 1-20　DengJiXiangXi

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	khid	int	4	0				是		
3	djid	int	4	0				是		

表 1-21　DengLuYongHu

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	uname	nvarchar	100	0				否		
3	password	nvarchar	100	0				否		
4	email	nvarchar	100	0				否		
5	phone	nvarchar	100	0				否		
6	yhlb	int	4	0				是		
7	fdid	int	4	0				是		

表 1-22　FenDian

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	fdname	nvarchar	100	0				否		
3	location	nvarchar	100	0				否		

表 1-23　KeFang

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	roomnum	nvarchar	20	0				否		
3	status	nvarchar	6	0				否		
4	fdid	int	4	0				是		
5	kflbid	int	4	0				是		

表 1-24　KeFangLeiBie

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	typename	nvarchar	100	0				否		
3	price	decimal	9	0				否		
4	bookrate	decimal	9	0				否		
5	hourprice	decimal	9	0				否		
6	pic	nvarchar	50	0				是		

表 1-25　KeHu

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	sid	nvarchar	19	0				否		
3	uname	nvarchar	100	0				否		
4	addr	nvarchar	600	0				否		
5	gender	nvarchar	6	0				否		
6	pwd	nvarchar	50	0				是		

表 1-26　sysdiagrams

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	name	sysname	256	0				否		
2	principal_id	int	4	0				否		
3	diagram_id	int	4	0	是	是		否		
4	version	int	4	0				是		
5	definition	varbinary	MAX	0				是		

表 1-27　TempKeHu

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0		是		否		
2	sid	nvarchar	19	0				否		
3	uname	nvarchar	100	0				否		

表 1-28　YongHuLeiBie

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	shortcode	nvarchar	60	0				否		
3	typename	nvarchar	100	0				否		

表 1-29　YouHuiHuodong

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	title	nvarchar	100	0				否		
3	cont	text	16	0				否		
4	begintime	nvarchar	60	0				否		
5	endtime	nvarchar	60	0				否		

表 1-30　YuDing

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	khid	int	4	0				是		
3	booktime	nvarchar	100	0				否		
4	begintime	nvarchar	100	0				否		
5	endtime	nvarchar	100	0				否		
6	isOver	bit	1	0				否		

表 1-31　YuDingXiangXi

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	ydid	int	4	0				是		
3	kfid	int	4	0				是		

表 1-32　ZhuBanXinXi

序号	列名	数据类型	长度	小数位	标识	主键	外键	允许空	默认值	说明
1	id	int	4	0	是	是		否		
2	yhhdid	int	4	0				是		
3	fdid	int	4	0				是		

项目 2　网站页面开发

21　项 目 引 入

李明是 Smart On Line 电子商城的一位开发人员，接到任务协助创建一个新的 B2C (Business to Customer)模式的 Smart On Line Web 应用程序，包括设计各种相关页面。要求电子商城应用程序实现广告轮显、菜单导航、所有页面风格一致等要求。

22　项 目 分 析

经过小组讨论和仔细分析，建议 Smart On Line 电子商城站点采用 Visual Studio .NET 2012集成开发环境，使用 ASP. NET 技术进行开发。“网站页面开发”子项目分解为 4 个任务进行。第一个任务是建立 Smart On Line 电子商城站点，添加网站图标等；第二个任务是设计 Smart On Line 站点母版页，实现站点页面风格统一；第三个任务则是实现网站的首页页面制作；第四个任务是实现首页广告轮显。

23　知 识 准 备

2.3.1　ASP. NET 处理过程及运行机制

开发 Smart On Line 项目之前，非常有必要对 ASP. NET 处理过程及运行机制作下具体的阐述，减少程序开发过程中碰到的疑惑。网站开发过程中碰到的第一个关键概念就是 HTTP(HyperText Transfer Protocol)，如图 2-1 所示。

1. 连接到 Web 服务器

HTTP 是计算机通过网络进行通信的协议，使 HTTP 客户(如 Web 浏览器)能够从 Web 服务器请求信息和服务。Web 浏览器与 Web 服务器连接需要完成下面 7 个步骤。

(1) 建立 TCP 连接。

在 HTTP 工作开始之前，Web 浏览器首先以 TCP 与 Web 服务器建立连接，该协议与 IP 共同构建 Internet，即著名的 TCP/IP 协议族，因此 Internet 又被称作 TCP/IP 网络。HTTP 是比 TCP 更高层次的应用层协议，根据规则，只有低层协议建立之后才能进行更高层协议的连接，因此首先要建立 TCP 连接，一般 TCP 连接的端口号是 80。

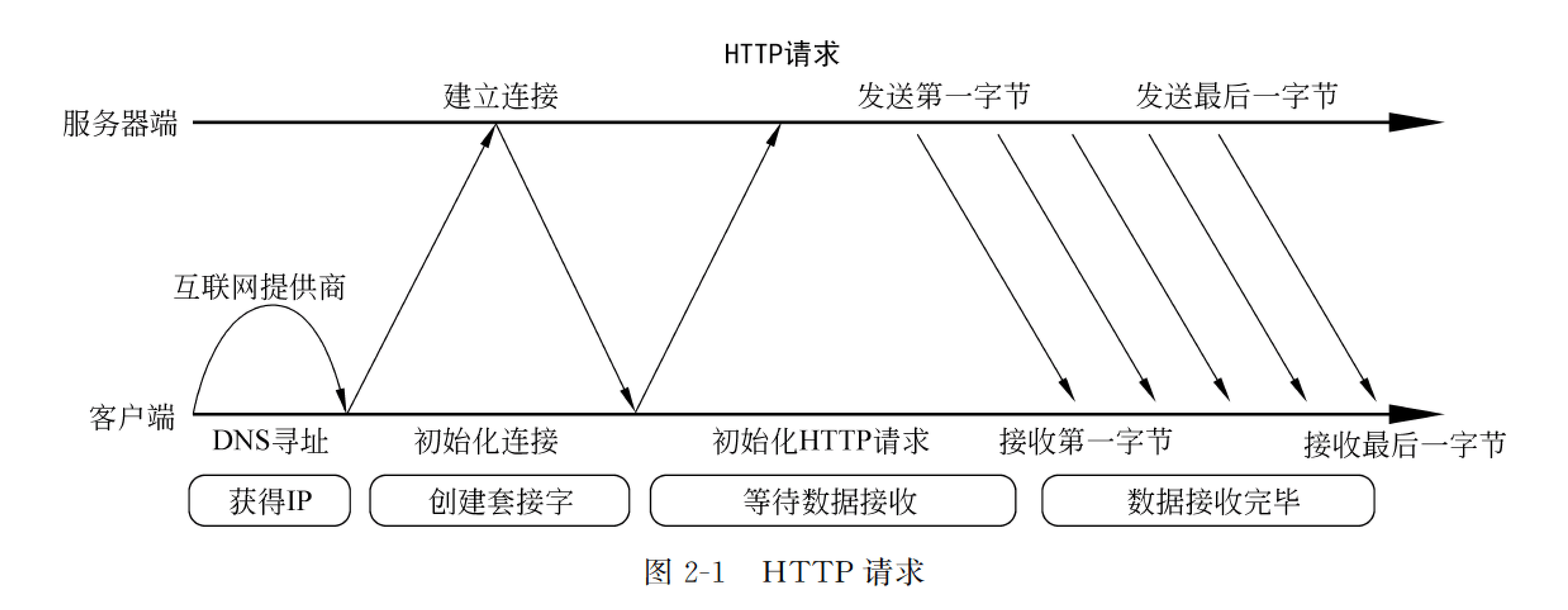


图 2-1　HTTP 请求

(2) Web 浏览器向 Web 服务器发送请求命令。

一旦建立了 TCP 连接，Web 浏览器就会向 Web 服务器发送请求命令。

(3) Web 浏览器发送请求头信息。

浏览器发送其请求命令之后，还要以头信息的形式向 Web 服务器发送一些其他的信息，之后浏览器发送一空白行来通知服务器，标识结束该头信息的发送。

(4) Web 服务器应答。

客户机向服务器发出请求后，服务器会向客户机回送应答“HTTP/1.1 200 OK”，应答的第一部分是协议的版本号和应答状态码。

(5) Web 服务器发送应答头信息。

发送关于它自己的数据及被请求的文档。

(6) Web 服务器向浏览器发送数据。

Web 服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此结束，接着它以 Content-Type 应答头信息所描述的格式发送用户所请求的实际数据。

(7) Web 服务器关闭 TCP 连接。

一般情况下，一旦 Web 服务器向浏览器发送了请求数据，它就要关闭 TCP 连接，然后如果浏览器或者服务器在其头信息加入了这行代码(Connection：keep-alive)，TCP 连接在发送后将仍然保持打开状态，于是浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，节约了网络带宽。

2. 访问 ASP.NET 页面

当通过浏览器访问一个 ASP.NET 页面时，通常要经过 8 个处理过程，如图 2-2 所示。

(1) 一个 Web 浏览器客户端发出一个 HTTP 请求访问.aspx 页面。

(2) IIS(Internet Information Services)接收请求并将其转交给 ASP.NET ISAPI 处理。

(3) ASP.NET 中的 HTTP 运行池解析该请求来决定调用合适的 HTTP 处理程序或者 HTTP 处理程序工厂。

(4) HTTP 模块执行请求预处理，包括高速缓存查询和授权等。

(5) HTTP 处理程序或者 HTTP 处理程序工厂被调用，进行与请求真正相关的处理工作。

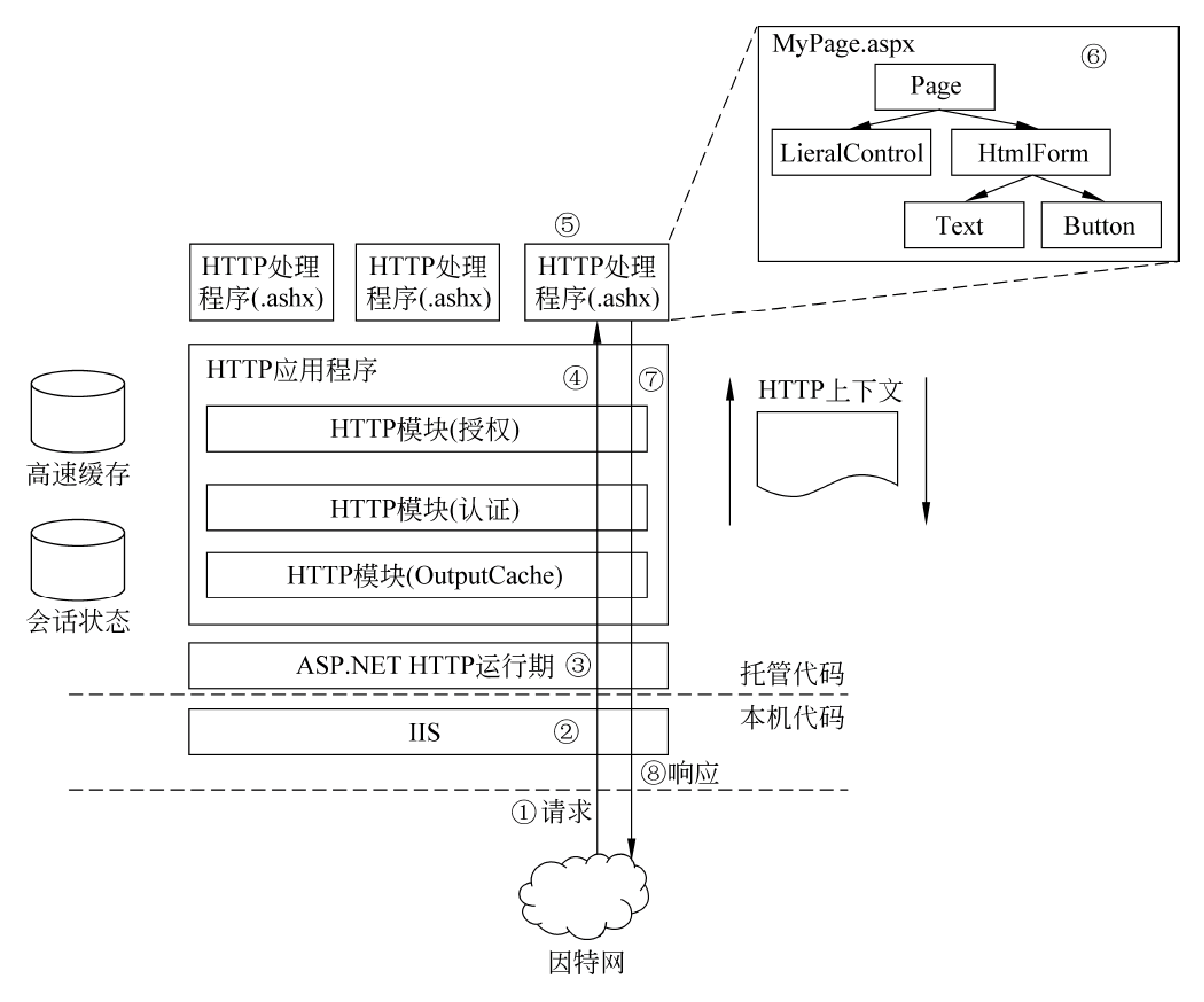


图 2-2　ASP. NET 请求处理过程

(6) 在. aspx 页面请求情况下,页面实例化一个控件树,同时显示该控件树来形成响应。

(7) HTTP 模块执行请求的后处理。

(8) 相应的结果发送回 Web 客户端。

从图 2-3 中不难发现,ASP. NET 页面运行的整体流程和一般处理程序一样,不同之处在于它调用 Page 类的 ProcessRequest 方法创建页面控件树,执行页面声明周期。其流程如下描述。

(1) 调用了 Page 类的 ProcessRequest 方法。

(2) 打造页面控件树(_buildControTree()方法)。

(3) 执行页面生命周期,方便程序员在事件中注册方法,实现自己的功能。

(4) 调用 Render 方法,生成 HTML 代码。

2.3.2　Visual Studio 2012 集成开发工具

在本项目中,使用微软的开发工具 VS2012(Visual Studio . NET 2012)进行代码演练。Visual Studio . NET 2012 是一个强大的集成开发工具,为开发提供了丰富的开发工具(如图 2-4 所示)。

VS2012 中重要的工具包括“窗体设计器”、“工具箱”、“解决方案资源管理器”、“属性窗口”、“对象浏览器”等,大多数工具可从“视图”菜单打开。表 2-1 描述了常用窗口及其对应功能。

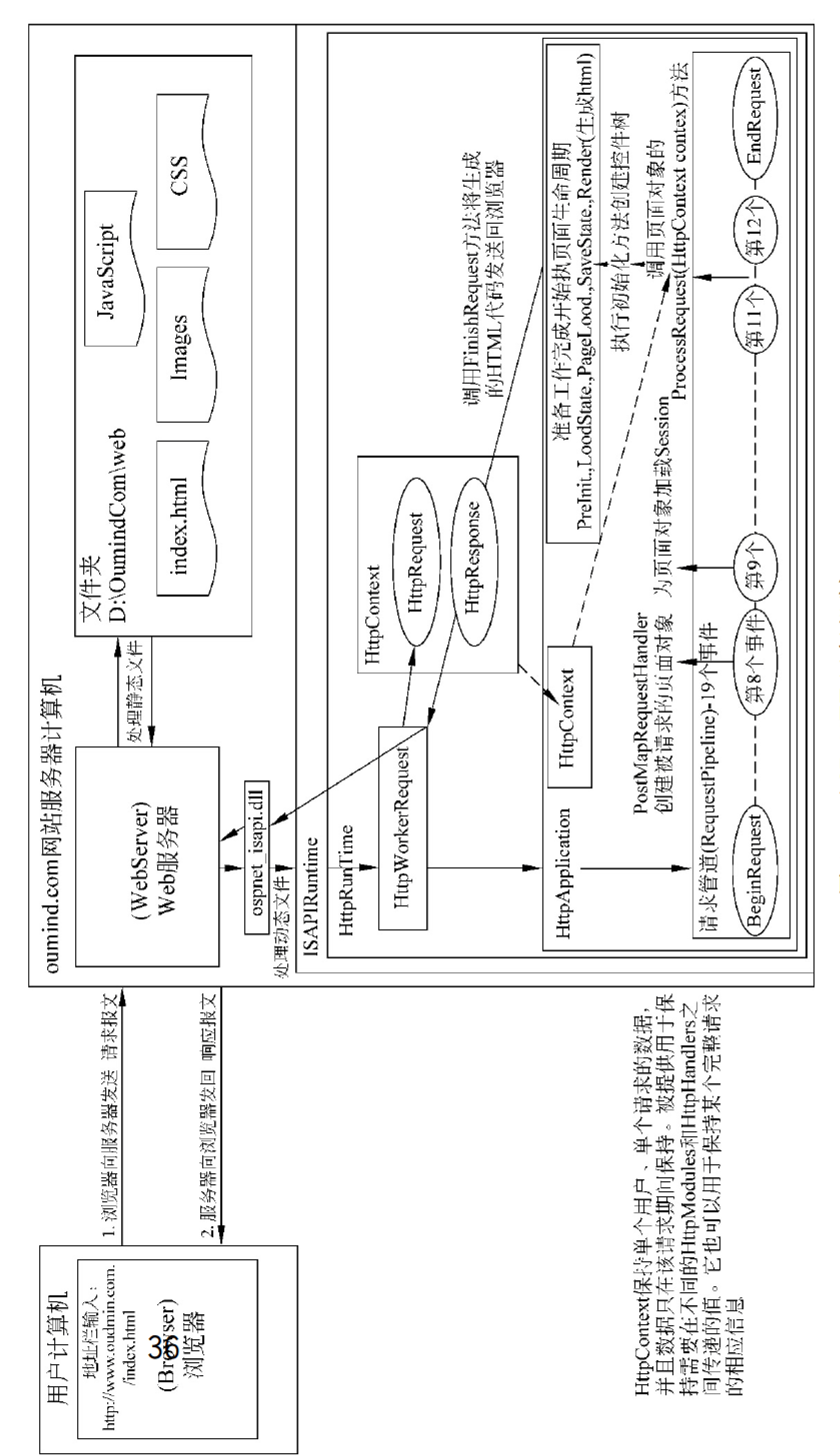


图 2-3 ASP. NET 运行机制

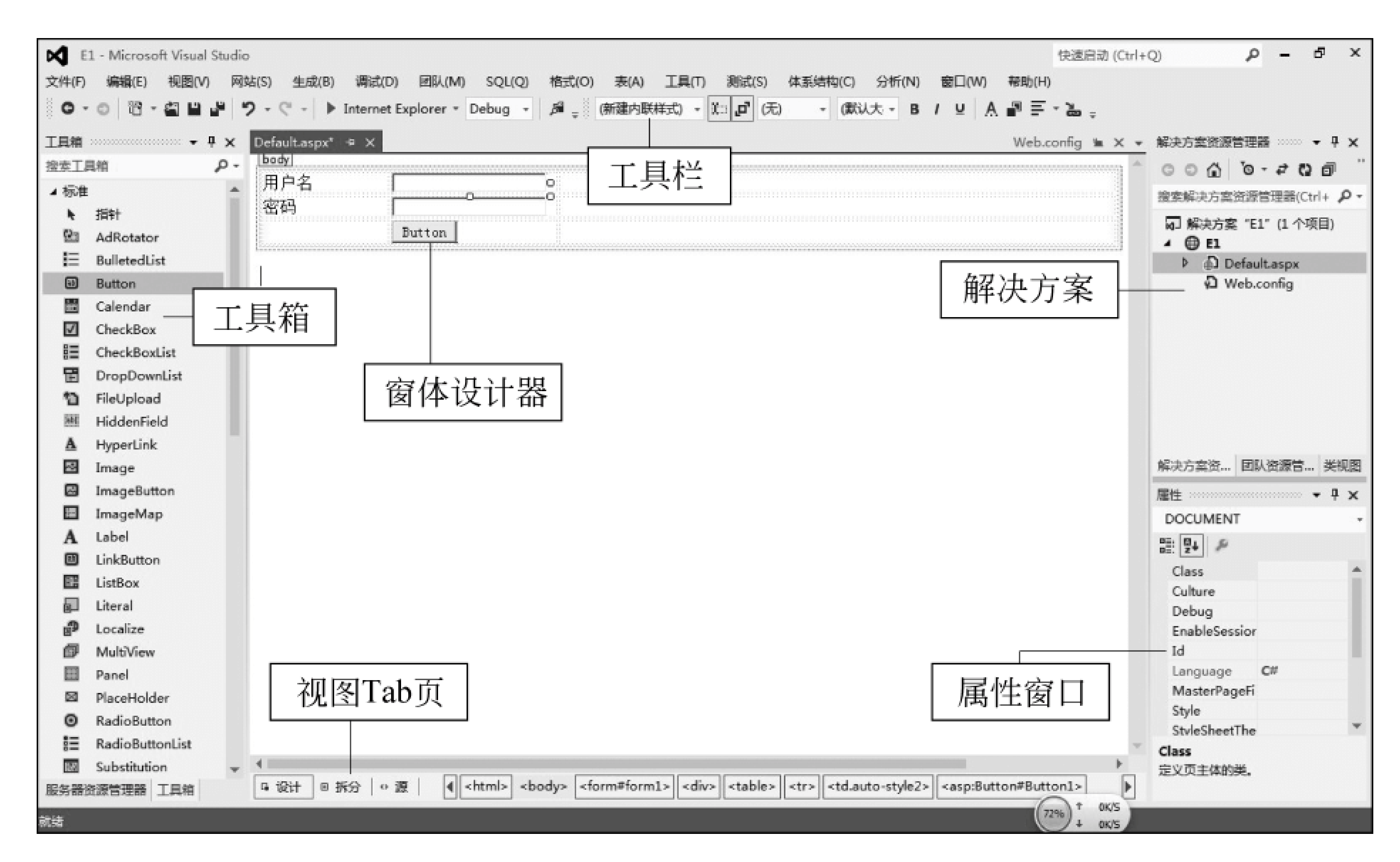


图 2-4　Visual Studio 2012 IDE 漫游

表 2-1　Visual Studio .NET 2012 集成开发环境常用窗口及功能

窗　口	用　途
工具箱	提供控件和 HTML 元素，可以直接拖放到页面上。工具箱的元素是按照功能划分的
工具栏	提供格式化文本、查找文本和其他命令。一些工具栏只会在使用设计视图时可用
解决方案管理器	显示 Web 站点上的文件和文件夹
属性窗口	允许更改页面、控件和其他对象属性
视图 Tab 页	提供统一文档的不同视图。设计视图是一种近乎所见即所得的编辑界面。源视图是这个页面的 HTML 编辑器

VS2012 集成开发环境提供了智能感知功能，用于在源代码视图中为用户提供有关服务器控件、HTML 控件和其他元素的句法信息。在编码时，不必让代码和文本编辑器或即时窗口、命令窗口执行语言元素搜索。直接向代码中插入语言元素，甚至可以通过智能感知直接完成输入工作。智能感知功能具体包括列出成员、显示参数信息两个常用的具体功能。

1．列出成员

输入触发器字符，将按类型(或命名空间)显示有效成员的列表，如图 2-5 所示。如果继续输入字符，则列表将会经过筛选变成仅包括这些字符开头的成员。在选择项后，可以通过

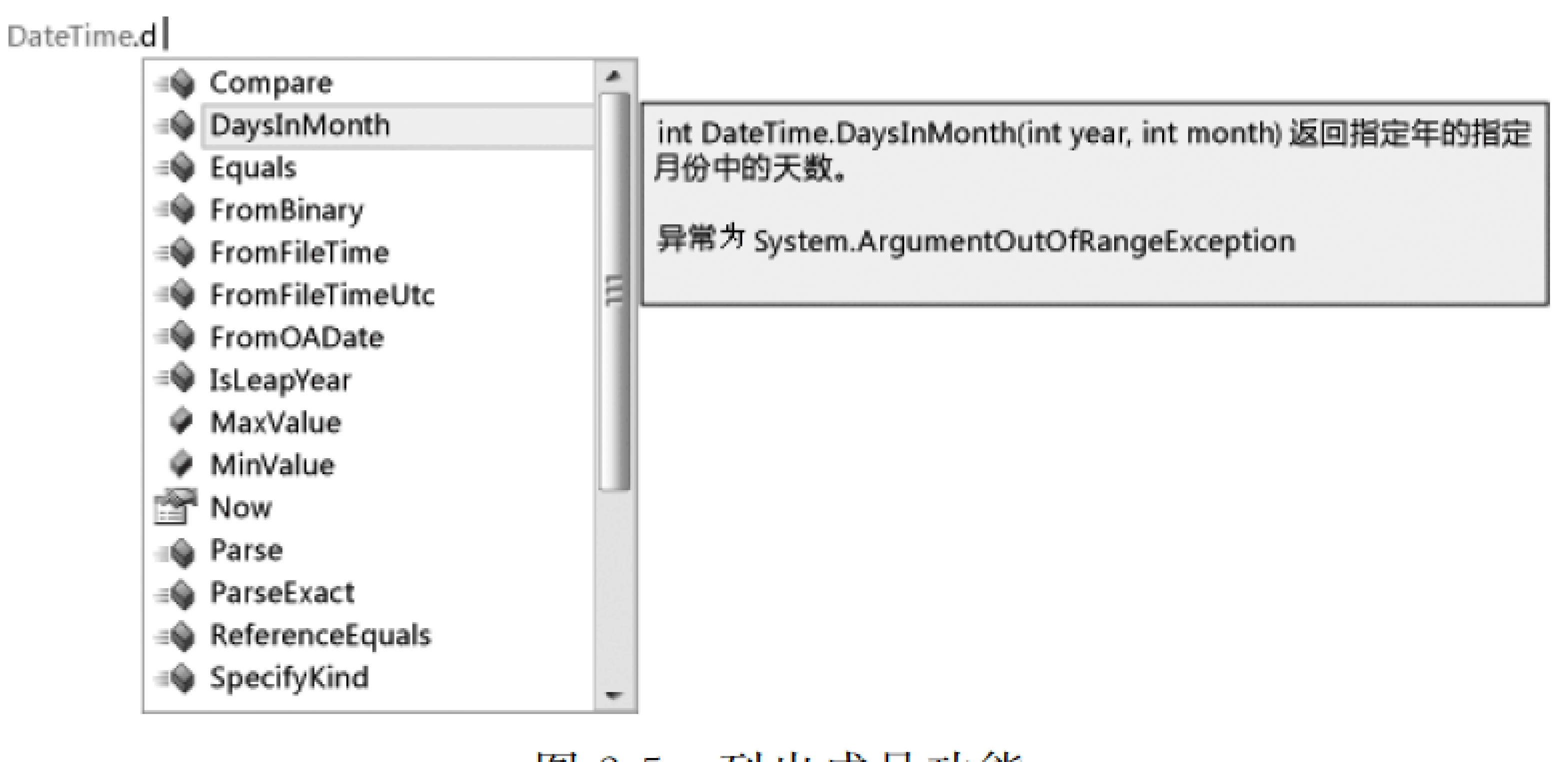


图 2-5　列出成员功能

按 Tab 键或 Space 键将其插入到代码中。如果选择一个项目并输入句点，将出现该项目，其后紧跟句点并带出另一个成员列表。在成员列表中，左边的图标表示成员的类型，如命名空间、类、函数或变量。列表较长时可以按 PageUp 和 PageDown 键来在列表中上下移动。

此外也可以通过按 Ctrl＋J 组合键、单击“编辑”→IntelliSense→“列出成员”命令或在编辑器工具栏上单击“列出成员”按钮来手动调用“列出成员”功能。在空行上或可识别范围之外调用列表时，将显示全局命名空间中的符号。

2. 显示参数信息

“显示参数信息”功能提供有关函数或特性所需的参数数目、参数名称和参数类型的信息、特性泛型类型参数或模板。当输入一个函数时，以粗体显示的参数表示输入函数所需的下一参数（如图 2-6 所示）。对于重载函数，可以按 PageUp 和 PageDown 键查看函数重载的其他参数信息。



图 2-6　显示参数信息

2.3.3　使用 Visual Studio 2012 创建 Web 站点、编辑页面

使用 VS2012 集成开发环境进行项目开发时，创建 Web 站点和编辑页面是常用的两项操作。下面就如何创建 Web 站点和编辑页面进行讲解。

1. 创建 Web 站点

当开发一个新的 Web 动态网站或信息系统时，首先需要创建一个新的网站。下面介绍创建新 Web 站点的详细步骤。

(1) 打开 Visual Studio 2012 .NET 集成开发环境，单击“文件”→“新建”菜单，然后选择“网站”命令。

(2) 在“新建网站”对话框中，单击“ASP. NET 空网站”选项。

(3) 在“位置”文本框中，输入所创建 Web 站点的路径和文件夹。可以是一个本地路径或者是网络计算机的 UNC 路径。

(4) 如果需要通过浏览查找已有的目录，请按照以下方法。

① 单击“浏览”按钮。

② 在“选择位置”对话框中，单击“文件系统”选项卡。

③ 在文件系统中，选择所想使用的文件夹或者在“文件夹”文本框中输入路径。

④ 单击“打开”按钮，返回“新建 Web 站点”对话框。

(5) 单击 OK 按钮。

通过上面 5 个步骤，就完成了 Web 站点的创建，并在解决方案管理器中显示相关文件夹。

注意：在 Visual Studio .NET 2012 旗舰版本中新建 Web 站点，不会生成默认页面，需要添加新的 Web 窗体页面（见添加新的 Web 窗体页面）。

2. 添加新的 Web 窗体页面

可以添加很多类型的页面和组件到 Web 站点中，包括简单的 HTML 页面、层叠样式表、Web 配置文件和 ASP. NET Web 窗体。假设需要创建一个新的用于显示相关信息的页面，添加窗体的步骤如下。

(1) 在解决方案资源管理器中，右击 Web 站点（例如 C:\Web），然后单击“添加新项”菜单项。

（2）在 Visual Studio 已安装的模板中，单击“Web 窗体”选项。

（3）在“名称”文本框中，输入 ProductInfo。

（4）单击“将代码放在单独的文件中”复选框，取消选择“将代码放在单独的文件中”选项。

在这个示例中，将创建一个包含代码和 HTML 的单文件页面。ASP. NET 提供两个用于管理可视元素和代码的模型，即单文件页模型和代码隐藏页模型。这两个模型功能相同，两种模型中可以使用相同的控件和代码。在单文件页模型中，页的标记及其编程代码位于同一个物理文件中(. aspx)。编程代码位于 Script 块中，该块包含 runat＝server 属性，此属性将其标记为 ASP. NET 应执行的代码。其中，Script 块可以包含 Web 页所需的任意多的代码。代码可以包含页中控件的事件处理程序、方法、属性及通常在类文件中使用的任何其他代码。在对该页进行编译时，编译器将生成和编译一个从 Page 基类派生或从使用@Page 指令的 Inherits 属性定义的自定义基类派生的新类。例如，如果在应用程序的根目录中创建一个名为 Login 的 ASP. NET 新网页，则随后将从 Page 类派生一个名为 ASP. Login_aspx 的新类。对于应用程序子文件夹中的页，将使用子文件夹名称作为生成的类的一部分。生成的类中包含. aspx 页中的控件的声明以及事件处理程序和其他自定义代码。在生成页之后，生成的类将编译成程序集，并将该程序集加载到应用程序域，然后对该页类进行实例化并执行该页类，以将输出呈现到浏览器。

在代码隐藏模型中，页的标记和服务器端元素(包括控件声明)位于. aspx 文件中，而页代码则位于单独的代码文件中。该代码文件包含一个分部类，类是使用 partial 关键字进行声明的，以表示该代码文件只包含构成该页的完整类的全体代码的一部分。而在页运行时，编译器将读取. aspx 页以及它在@Page 指令中引用的文件，将它们汇编成单个类，然后将它们作为一个单元编译为单个类。在分部类中，添加应用程序要求该页所具有的代码。此代码通常由事件处理程序构成，但是也可以包括需要的任何方法或属性。

3. 添加控件

VS2012 提供了丰富的控件，帮助开发者设计良好的用户界面。VS2012 中向 Web 页面添加控件可以通过两种方式：一是选择某种控件，拖放控件到页面；二是双击控件，自动将控件添加到页面特定位置。

下面通过“单击按钮，在标签中显示文本框中的姓名”简单功能来描述和演示如何向 Web 窗体中添加控件和设置属性，界面如图 2-7 所示。下面步骤描述如何添加 TextBox、



图 2-7　显示输入框中的姓名的简单功能

Label 和 Button 控件到页面中。

添加控件到页面步骤的简单描述如下。

(1) 单击“设计”选项卡，切换到“设计视图”。

(2) 从“标准”组中拖放三个控件到页面上：一个 TextBox 控件，一个 Button 控件和一个 Label 控件，ID 分别为 TextBox1、Button1 和 Label1。

(3) 单击页面，在页面左端输入“用户名”文本作为 TextBox 控件的标签。ASP. NET 允许在同一个页面上混合使用静态 HTML 和服务器控件。

4. 设置控件属性

VS2012 提供多种方式来设置页面中控件的属性。在本节中，通过在“设计”和“源”视图下设置按钮文本和颜色，讲解属性设置的两种常用途径。其基本步骤如下。

(1) 单击 Button 控件，然后在“属性窗口”中的 Text 属性框中输入“显示姓名”。

(2) 单击“源”选项卡。

这时将以“源”视图方式显示页面的 HTML。VS2012 提供了“设计”、“源”和“拆分”三种视图方式。“设计”视图使用一种近似所见即所得的视图来显示 ASP. NET 网页、母版页、内容页、HTML 页和用户控件。通过“设计”视图可以对文本和元素进行以下操作：添加、定位、调整大小以及使用特殊菜单或“属性”窗口设置其属性。“设计”视图只显示文档的正文，即显示<body>和</body>标记之间的部分。虽然使用“文档属性”窗口可以编辑 head 元素的某些属性(如文档的标题)，但必须切换到“源”视图，才能编辑不在 body 元素内的那些元素的属性。“源”视图显示所有的 HTML 元素和脚本，包括 VS2012 为服务器控件创建的元素。控件用类似 HTML 的句法描述，除了标签首部“asp:”和属性 runat＝server，其余则作为 HTML 属性来声明。

注意：控件包含在<form>元素中，该元素同样具有“runat＝server”属性。runat＝server 属性和控件标签的首字符“asp:”标识使得控件在页面运行时被 Web 服务器上的 ASP. NET 处理。在<form>和<script runat＝server>元素外的代码会被浏览器当作客户端代码。

(3)“源”视图方式下，单击标签<asp:Label>后面的空白处，然后按 space 键，系统将弹出一个下拉菜单，显示可以为 Label 控件设置的属性。

(4) 弹出的下拉属性列表中，单击 ForeColor 列表项，接着输入等号，智能感知将显示出可用的颜色列表。在任何时候，按 Ctrl＋J 组合键即可实现智能感知下拉菜单。

(5) 单击列表项(例如 Blue)，将 ForeColor 属性修改为已选的颜色值。

2.3.4 使用母版页统一页面风格

使网站有统一的外观和操作方式是开发人员一直试图解决的一个问题。大多数网站都有标题、脚标和菜单结构，但它们对放置其中的各个项目的处理方式不同。在 ASP. NET 中，用户控件(和 CSS)提供了使网站布局保持一致的最佳方案，因为它们是面向对象的，可以定义一个或多个页面上的通用标题、脚标和菜单。但是，大多数开发人员都要在每个页面上使用 Reference 指令以及相关的 TagPrefix 和 TagName 属性来引用用户控件，导致代码的重复。为了解决这些重复代码的问题，一些开发人员尝试把用户控件嵌入其他用户控件中。这个方式的确有效，但在加载或回送操作中需要编程访问控件时，嵌套的对象模型很难

使用。ASP. NET 为网站提供统一的布局和结构。该方式不是采用嵌套的用户控件、定制类和模板，而是使用 ASP. NET 的母版页。通过使用母版页技术定义网站的整体布局，且所花的时间和精力非常少。它们可以在一个地方定义，在整个网站上共享，且不需要把标题、菜单和用户控件硬编码到每个页面上。

1. 母版页基础

母版页提供了一种简单而高效的模板框架，它允许在一个模板文件中定义网站的公共组件，例如菜单、标题和脚标。这将快速而轻松地修改网站的整体外观、操作方式和布局，因为对母版页模板的修改会自动应用于整个网站。许多网站都只需要一个母版页，但也可以根据需要创建多个母版页——一个用于主页、一个用于子页面。母版页也可以根据用户的配置或其他条件动态加载，因此很容易提供页面的可打印版本，而无须修改页面的内容。所有的母版页都有. master 扩展名，它映射到处理程序 System. Web. HttpForbidden Handler 上。这个映射在 Web 服务器的默认 web. config 文件中定义，可以防止母版页直接在浏览器中打开，就好像带. ascx 和. config 扩展名的文件不能用浏览器查看一样。试图在浏览器中查看扩展名为. master 的文件，会导致一个“这类页面不能打开”的错误。母版页依赖. NET Framework中的已有类来执行其逻辑。它们直接派生自 UserControl 类：

```
public class MasterPage: System.Web.UI.UserControl
```

派生自 UserControl 类的结果是，母版页有一组标准的属性，例如 Page、Session、Request 和 Response，它们可以以编程方式访问。母版页不能直接在 Internet Information Services(IIS)应用程序(如用户控件)之间共享，否则会生成一个错误。简单的母版页(AdminMP. master)如示例 2-1(AdminMP. master)所示。

示例 1：简单母版页(AdminMP. master)。

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="AdminMP.master.cs" Inherits="Admin_AdminMP"
%>
<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1
-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="../css/Admin.css" rel="Stylesheet" />
</head>
<body>
    <form id="form1" runat="server">
    <div id="containerdiv">
        <div id="headerdiv">
        </div>
        <div id="navdiv">
            <asp:TreeView ID="TreeView1" runat="server" BorderColor="
            #CC0000"
                BorderStyle="Solid" BorderWidth="1px" ShowLines="True">
                <HoverNodeStyle BackColor="#993366" />
                <Nodes>
                    <asp:TreeNode
                        NavigateUrl="~/Admin/Level1Management/InsertLevel1.aspx"
```

```
                        Text="添加一级目录" Value="添加一级目录"></asp:TreeNode>
                    <asp:TreeNode
                        NavigateUrl="~/Admin/Level1Management/EditLevel1.aspx"
                        Text="编辑一级目录" Value="编辑一级目录"></asp:TreeNode>
                    <asp:TreeNode
                        NavigateUrl="~/Admin/Level2Management/Level2Insert.
                        aspx"
                        Text="添加二级目录" Value="添加二级目录"></asp:TreeNode>
                    <asp:TreeNode
                        NavigateUrl="~/Admin/Level2Management/EditLevel2.aspx"
                        Text="编辑二级目录" Value="编辑二级目录"></asp:TreeNode>
                    <asp:TreeNode
                        NavigateUrl="~/Admin/Level3Management/InsertLevel3.
                        aspx"
                        Text="添加三级目录" Value="添加三级目录"></asp:TreeNode>
                    <asp:TreeNode
                        NavigateUrl="~/Admin/Level3Management/EditLevel3.aspx"
                        Text="编辑三级目录" Value="编辑三级目录"></asp:TreeNode>
                    <asp:TreeNode NavigateUrl="~/Admin/AddItem.aspx"
                        Text="添加商品" Value="添加商品">
                    </asp:TreeNode>
                    <asp:TreeNode Text="编辑商品" Value="编辑商品"></asp:TreeNode>
                </Nodes>
                <NodeStyle ImageUrl="~/WebIcon/Green Ball.png" />
                <SelectedNodeStyle ImageUrl="~/WebIcon/Blue Ball.png" />
            </asp:TreeView>
        </div>
        <div id="contentdiv">
            <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
                放置正文内容
            </asp:ContentPlaceHolder>
        </div>
        <div id="footdiv">
            CopyRight: NBCC</div>
    </div>
    </form>
</body>
</html>
```

上面示例 1 中的代码定义了 TreeView、网站 Banner 和脚标在页面整个结构中的位置，还使用 ContentPlaceHolder 服务器控件定义了页面内容在母版页中的位置。ContentPlaceHolder 派生自 Control，它并不添加自己的属性、方法或事件。其唯一作用是标记页面内容在母版页模板中的位置。这里的母版页只有一个 ContentPlaceHolder 控件，但如果页面的内容应该位于不同的区域（例如左列和右列），就可以添加多个 ContentPlaceHolder 控件。添加多个 ContentPlaceHolder 控件时，需要确保每个控件都有唯一的 ID。母版页与普通的.aspx 源代码非常相似，例如，包括<html>、<body>、<form>等 Web 元素，但是，与普通页面还是存在差异。差异主要有两处(粗体代码所示)。差异一是代码头不同，母版页使用的是 Master，而普通.aspx 文件使用的是 Page。除此之

外，二者在代码头方面是相同的。差异二是母版页中声明了控件 ContentPlaceHolder，而在普通.aspx 文件中是不允许使用该控件的。

特别需要注意的是，在母版页中定义的图像或其他资源的路径与使用母版页显示的实际页面相关，该路径与母版页本身的位置没有关系。例如，如果一个页面位于网站的根目录下，且存在一个 Images 子文件夹，则使用 Images/ImageName.gif 定义母版页中的图像就是正确的，因为 Images 子文件夹位于该页面的下一级。对于不位于网站根目录的页面，该路径不起作用，最终会导致图像在浏览器中由于找不到源文件而不显示。应该把母版页的路径定义为当前网站根目录中的图像、样式表、JavaScript 文件等。对于 ASP.NET 服务器控件，如 Image 控件，可使用～字符表示图像路径从网站的根目录开始。母版页允许采用页面代码分离方式，以使 HTML 代码和编程代码清晰地分隔开来，就像 ASP.NET Web 窗体和用户控件那样。示例 1 中的母版页示例与一个 C＃类 AdminMP 关联起来，该类位于 AdminMP.master.cs 文件中。标准事件处理程序，如 Page_Load、Page_PreRender 等都可以包含进来，以便在运行期间动态修改母版页的内容。

2. 创建和使用母版页

母版页中包含的是页面公共部分，即网页模板。因此，在创建示例之前，必须判断哪些内容是页面公共部分，这就需要从分析页面结构开始。通过分析可知，该页面（设名称为 Index.aspx）的结构如图 2-8 所示。

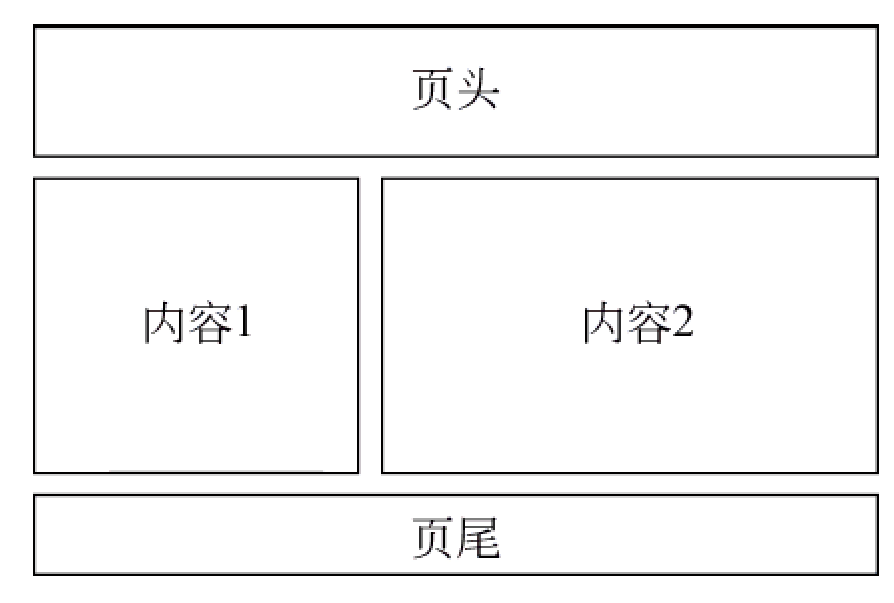


图 2-8　页面结构图

Index.aspx 页面由 4 个部分组成：页头、页尾、内容 1 和内容 2。其中页头和页尾是 Index.aspx 所在网站中页面的公共部分，网站中许多页面都包含相同的页头和页尾。内容 1 和内容 2 是页面的非公共部分，是 Index.aspx 页面所独有的。结合母版页和内容页的有关知识可知，如果使用母版页和内容页来创建页面 Index.aspx，那么必须创建一个母版页 MasterPage.master 和一个内容页 Index.aspx。其中母版页包含页头和页尾等内容，内容页中则包含内容 1 和内容 2。使用 Visual Studio 2012 创建母版页的步骤较简单，在“解决方案资源管理器”中右击网站，选择“选择”→“添加新项”命令，打开“添加新项”对话框（如图 2-9 所示）。

前面介绍了母版页的基本内容，接下来讲解如何使用母版页。要在内容页面中引用母版页，可以使用 Page 指令的 MasterPageFile 属性。MasterPageFile 属性中定义的路径应从 Web 应用程序的根目录开始，这样，如果页面移动到网站文件夹结构的另一级上，它仍包含指向母版页的正确路径。注意，内容页面不包含任何 HTML 或 body 标记，因为这些标记都是在母版页文件中定义的。内容页面的所有内容都必须放在一个或多个 Content 服务器控件中，所以一般看不到 Title 标记，也没有要放在该标记中的 head 标记。Page 指令包含一个 Title 属性，它可以用于设置内容页面的标题，否则内容页面会使用母生版页的标题作为自己的标题。标题也可以通过编程方式进行修改。

```
<%@ Page Language="C#" MasterPageFile="AdminMP.master" Title="编辑一级目录" %>
```

通常使用母版页存在两种应用场景，一种是需要将现有的 Web 页面从某个母版页中继



图 2-9　添加母版页

承，而另一种则是新建继承自某个母版页的 Web 页面。下面通过两个示例来详细介绍如何使用母版页。

示例 2：修改已存在的 Web 页面，使其从母版页中继承。

项目开发前期已经存在 EditLevel1.aspx 页面，现在开发小组重新设计了一个母版页(AdminMP.master)，要求将 EditLevel1.aspx 页面使用新设计的母版页 AdminMP.master。该示例则可通过以下步骤完成。

① 单击“源”按钮，切换到源代码视图。

② 在 Page 指令中输入 MasterPageFile=“~/Admin/AdminMP.master”。

③ 在 Page 指令后输入：

```
<asp:Content ID="Content1"
    ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

④ 将原先<form>标记间的内容复制到<asp:Content></asp:Content>间，最终源码如下所示。

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="EditLevel1.aspx.cs" Inherits="Admin_
Level1Management_EditLevel1" MasterPageFile="~/Admin/AdminMP.master"%>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <div>
        <asp:GridView ID="gvLevel1" runat="server" AutoGenerateColumns="False"
            CellPadding="4" ForeColor="#333333" GridLines="None" Width="614px"
            onrowcancelingedit="gvLevel1_RowCancelingEdit"
            onrowediting="gvLevel1_RowEditing"
                onrowupdating="gvLevel1_RowUpdating">
            <AlternatingRowStyle BackColor="White" />
            <Columns>
                <asp:BoundField DataField="Level1_ID" HeaderText="序号"
```

```
                ReadOnly="True" />
            <asp:BoundField DataField="Level1_Name" HeaderText="名称"
                ReadOnly="True" />
            <asp:TemplateField HeaderText="备注">
                <EditItemTemplate>
                    <asp:TextBox ID="TxtLevel1Desc" runat="server"
                        Text='<%#Bind("Level1_Desc") %>'></asp:TextBox>
                </EditItemTemplate>
                <ItemTemplate>
                    <asp:Label ID="Label1" runat="server"
                        Text='<%#Bind("Level1_Desc") %>'></asp:Label>
                </ItemTemplate>
            </asp:TemplateField>
            <asp:CommandField ShowEditButton="True" />
        </Columns>
        <EditRowStyle BackColor="#2461BF" />
        <FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor=
        "White" />
        <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor=
        "White" />
        <PagerStyle BackColor="#2461BF" ForeColor="White"
                HorizontalAlign="Center" />
        <RowStyle BackColor="#EFF3FB" Font-Size="Small"
            HorizontalAlign="Center" />
        <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True"
            ForeColor="#333333" />
        <SortedAscendingCellStyle BackColor="#F5F7FB" />
        <SortedAscendingHeaderStyle BackColor="#6D95E1" />
        <SortedDescendingCellStyle BackColor="#E9EBEF" />
        <SortedDescendingHeaderStyle BackColor="#4870BE" />
    </asp:GridView>
</div>
</asp:Content>
```

示例 3：新建 Web 页面，使其从母版页中继承。

开发小组要求后续创建的页面文件都要求从 AdminMP.master 母版页中继承，实现统一风格，例如编辑订单页面 EditOrder.aspx。则该示例可通过以下步骤实现。

① “解决方案资源管理器”窗口中，右击存放母版页的文件夹（例如 Admin 文件夹），单击“添加新项”菜单项，打开添加新项对话框，如图 2-10 所示。

② 在“名称”文本框中输入 EditOrder.aspx，选择“选择母版页”复选框，单击“添加”按钮，打开“选择母版页”对话框，如图 2-11 所示。单击 AdminMP.master 文件，再单击“确定”按钮。

2.3.5 使用样式表控制页面

CSS 即层叠样式表（Cascading StyleSheet），在网站开发时采用层叠样式表技术，可以有效地对页面的布局、字体、颜色、背景和其他效果实现更加精确的控制。只要对相应的代码做一些简单的修改，就可以改变同一页面的不同部分或者页数不同的网页的外观和格式。

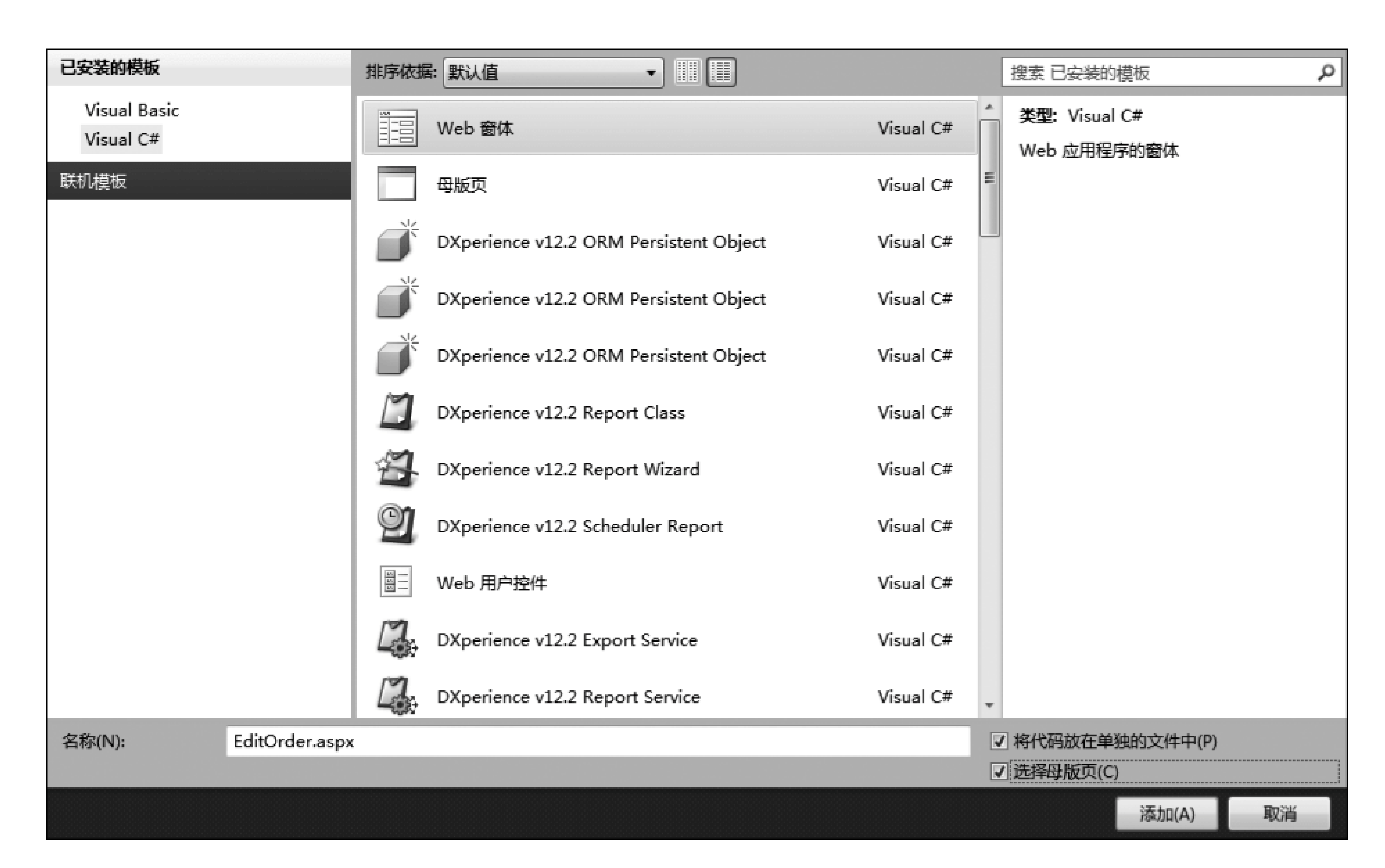


图 2-10　添加新项对话框

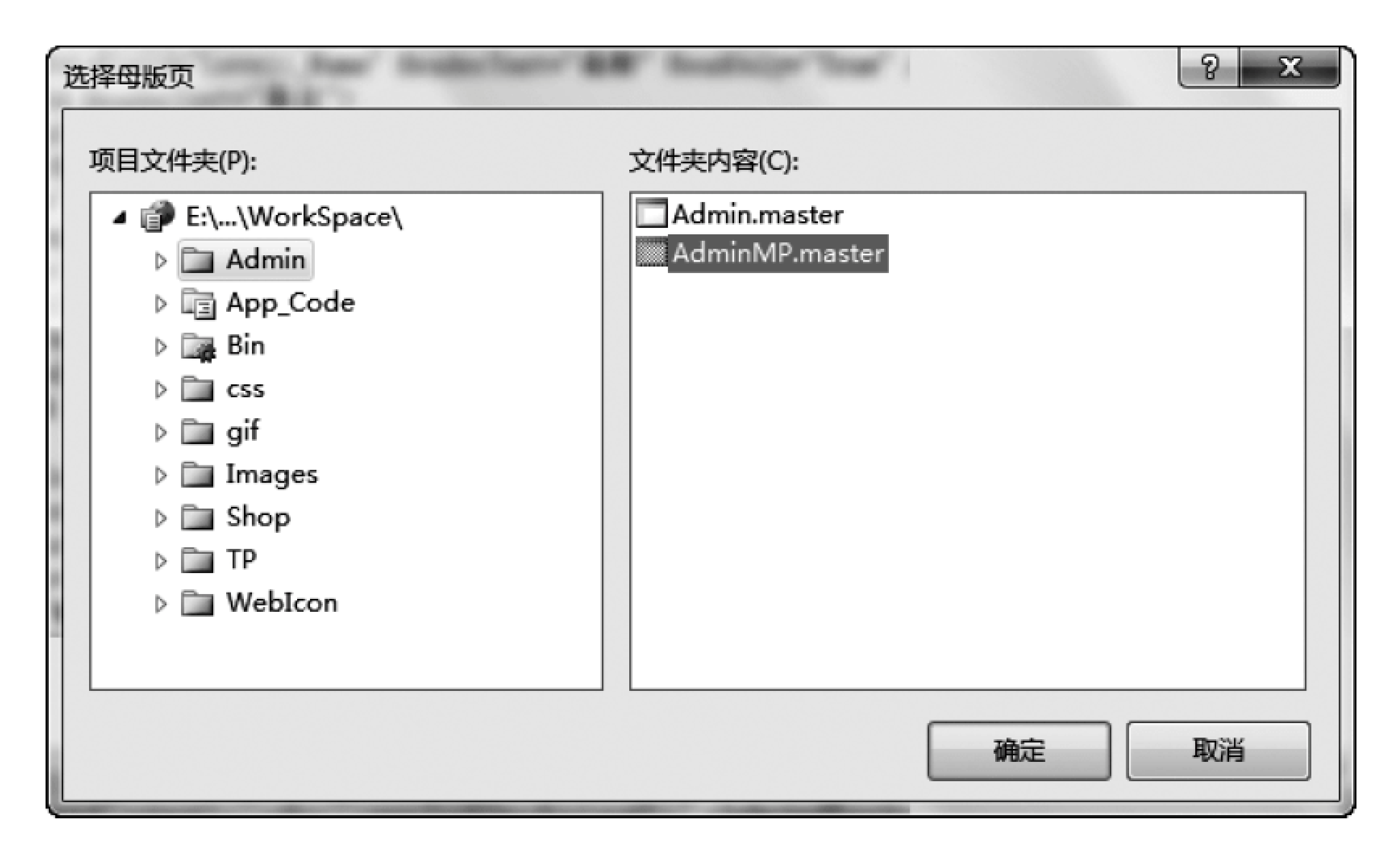


图 2-11　“选择母版页”对话框

结构和表现相分离，这也用 CSS 布局的特色所在。

1. CSS 盒子模型的组成

那么它为什么叫盒子呢？先说说我们在网页设计中常听的属性名：内容（content）、填充（padding）、边框（border）、边界（margin），CSS 盒子模式都具备这些属性，如图 2-12 所示。

可以把它想象成现实中上方开口的盒子，然后从正上方向下俯视，边框相当于盒子的厚度，内容相对于盒子中所装物体的空间，而填充相当于为防震而在盒子内填充的泡沫，边界相当于在这个盒子周围要留出一定的空间以方便取出其中的内容。所以整个盒模型在页面中所占的宽度是由“左边界＋左边框＋左填充＋内容＋右填充＋右边框＋右边界”组成的，而 CSS 样式中 weight 所定义的宽度仅仅是内容部分的宽度，这是许多初学者容易搞混的地方。

2. CSS 作用原理

CSS 的定义是由三个部分构成的：选择符（selector），属性（properties）和属性的取值

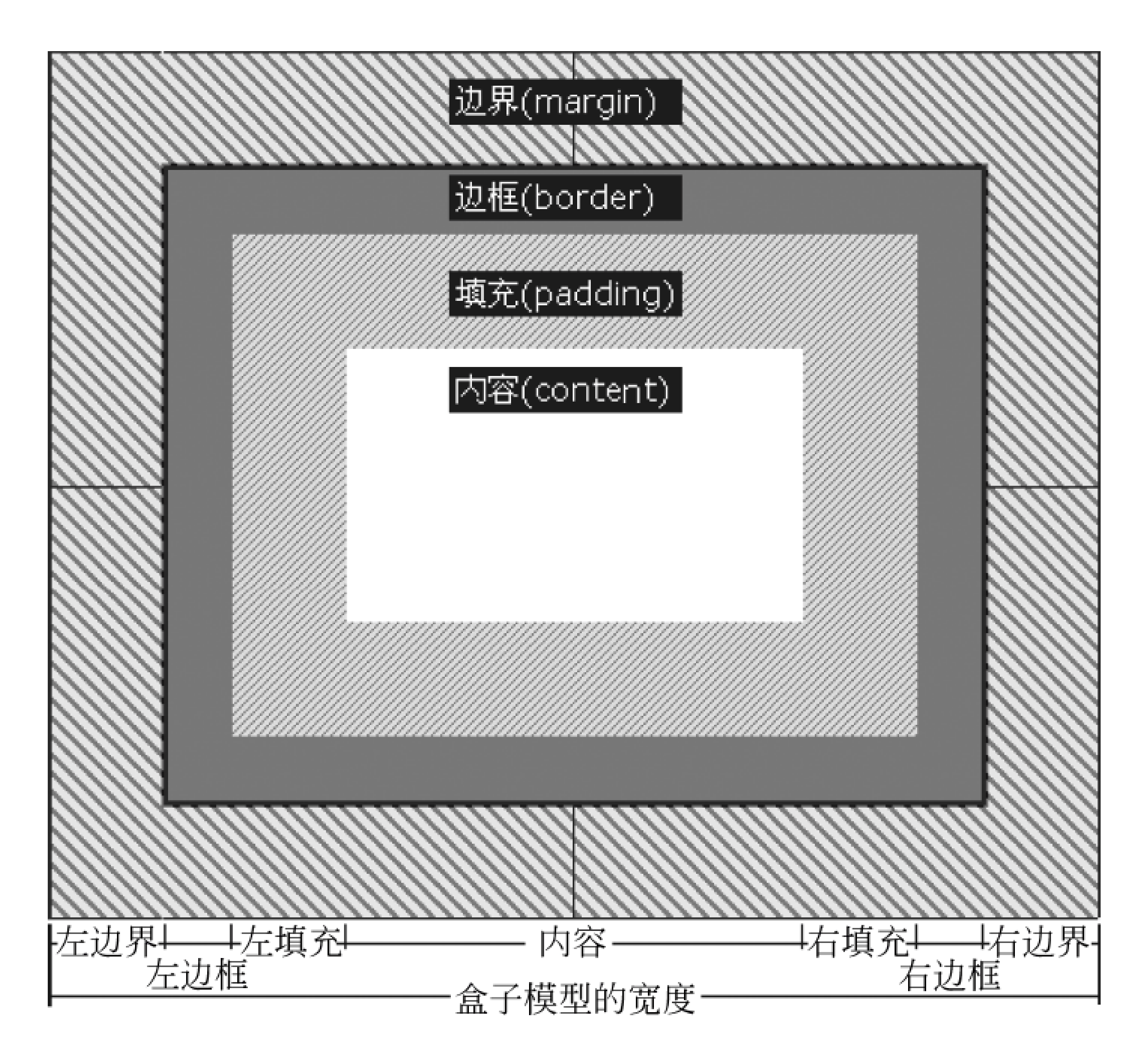


图 2-12　盒子模型

(value)。基本格式如下：selector{property：value}(选择符{属性：值})。选择符可以是多种形式，一般是要定义样式的 HTML 标记，例如 body、p、table 等，通过此方法定义它的属性和值，属性和值要用冒号隔开。

```
body{color: black}
```

选择符 body 是指页面主体部分，color 是控制文字颜色的属性，black 是颜色的值，此例的效果是使页面中的文字颜色变为黑色。如果属性的值由多个单词组成，则必须在值上加引号，比如字体的名称经常是几个单词的组合：p{font-family："sans serif"}(定义段落字体为 sans serif)。如果需要对一个选择符指定多个属性时，使用分号将所有的属性和值分开。

```
p{text-align: center; color: red}
```

此例的效果是段落居中排列，并且段落中的文字为红色。为了使定义的样式表方便阅读，建议采用分行的书写格式。

CSS 选择符：CSS 样式的名字，当在 HTML 文档中表现一个 CSS 样式的时候，就要用到一个 CSS。怎么用呢？这时就通过 CSS 选择符(CSS 的名字)来指定 HTML 标签使用此 CSS 样式。比如：

```
p
{
    font-size:12px;
}
.dreamdublue
{
    color:blue;
}
.dreamdu18px
```

```
{
    font-size:18px;
}
#dreamdured
{
    color:red;
}
```

在这段代码中，p、dreamdublue、dreamdured 都是选择符。CSS 选择符可以分为很多类，比如：类型选择符、id 选择符、class 选择符、通用选择符、分组选择符、包含选择符、元素指定选择符等，以下将一一介绍。

① 类型选择符。网页元素本身，定义时直接使用元素名称。比如：

```
Div{
    Width:774px; /* 把所有的 div 元素宽度定义为 774 像素 */
}
```

② id 选择符。它是唯一的，不同元素的 id 值是不能重复的，id 选择符为每个元素的具体对象定义不同的样式，方便用户更加精细地控制页面。使用 id 选择符时要先为每个元素定义一个 id 属性。使用 id 选择符时，需要使用＃进行标识。

```
<div id="top"></div>

#top{
    Width:774px; /* 把所有的 div 元素定义为宽度为 774 像素 */
}
```

③ class 选择符。在一个文档中可以为不同类型的元素定义相同的类名，class 可以实现把相同样式的元素统一为一类，使用 class 选择符时要先为每个元素定义一个 class 属性。使用 class 选择符时，需要使用英文点(.)进行标识。

```
<div class="red"></div>
<span class="red"></span>
<p class="red"></p>

.red{
    Color:red ;
}
```

④ 通用选择符。一种特殊的选择符，它用＊表示，是一个实用但又容易忽略的选择符。

```
* {
    font-size:12pt;/* 定义文档中所有字体大小为 12pt */
}
```

⑤ 分组选择符。不是一种选择符类型，而是一种选择符方法。当多个对象定义了相同的样式时，可以把它们分为一组。这样能够简化代码读/写，例如：

```
.class1{
    font-size:13px;
    color:red;
```

```
    text-decraotian:underline;
}
.class2{
    font-size:13px;
    color:blue;
    text-decroation:underline;
}
```

可以分组为

```
.class1,class2{
    font-size:13px;
    text-decroation:underline;
}
.class1{
    color:red;
}
.class2{
    color:blue;
}
```

使用分组有 2 原则：(1)方面原则；(2)就近原则(如果几个元素相邻，在一个模块内可以考虑使用分组选择符)。

⑥ 包含选择符。最有用的一类选择符，能够简化代码，实现大范围的样式控制。例如：

```
.div1 h2{
    /* 定义类的 div1 层中所有 h2 的标题样式 */
    font-size:18px;
}
.div1 p{
    /* 定义类的 div1 层中所有 p 的标题样式 */
    font-size:12px;
}
```

⑦ 元素指定选择符。有时候我们希望控制某种元素在一定范围内对应对象的样式，这时可以把 class 或 id 选择符组合起来使用。例如：

```
span.red{
/* 定义 span 中 class 为 red 的元素颜色为红色 */
    color:red;
}
div#top{
    /* 定义 div 中 id 为 top 的元素宽度为 100% */
    width:100%;
}
eg:
<div>
<h2 class="news"><h2>
<p class ="news"></p>
<span class="news"></span>
</div>
```

在上面代码中要使用 news 选择符显然不行，直接使用 p、h2 类型选择符也不太好，这时便可以使用元素指定选择符：

```
p.news{} h2.news{} span.news{}
```

3. 调用 CSS 样式

前面讲解了 CSS 基本语法，但要想在浏览器中显示出效果，就要让浏览器能识别并调用。当浏览器读取样式表时，要依照文本格式来读，这里介绍四种在页面中插入样式表的方法：链入外部样式表、内部样式表、导入外部样式表和内嵌样式。

① 链入外部样式表。链入外部样式表是把样式表保存为一个样式表文件中，然后在页面中用<link>标记链接到这个样式表文件，这个<link>标记必须放到页面的<head>区内，例如：

```
<head>
<link href="mystyle.css" rel="stylesheet" type="text/css" media="all">
</head>
```

上面这个例子中表示浏览器从 mystyle.css 文件中以文档格式读出定义的样式表。rel="stylesheet"是指在页面中使用这个外部的样式表。type="text/css"是指文件的类型是样式表文本。href="mystyle.css"是文件所在的位置。media 是选择媒体的类型，这些媒体包括：屏幕、纸张、语音合成设备、盲文阅读设备等。一个外部样式表文件可以应用于多个页面。当改变这个样式表文件时，所有页面的样式都随之而改变。在制作大量相同样式页面的网站时非常有用，不仅减少了重复的工作量，而且有利于以后的修改、编辑，浏览时也减少了代码的重复下载。样式表文件可以用任何文本编辑器（例如：记事本）打开并编辑，一般样式表文件扩展名为.css。内容是定义的样式表，不包含 HTML 标记。mystyle.css 这个文件的内容如下。

```
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
/* 定义水平线的颜色为土黄；段落左边的空白边距为 20 像素；页面的背景图片为 images 目录下的 back40.gif 文件 */
```

② 内部样式表。内部样式表是把样式表放到页面的<head>区里，这些定义的样式就应用到页面中了，样式表是用<style>标记插入的，从下例中可以看出<style>标记的用法。

```
<head>
⋮
<style type="text/css">
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
</style>
⋮
</head>
```

注意：有些低版本的浏览器不能识别 style 标记，这意味着低版本的浏览器会忽略 style 标记里的内容，并把 style 标记里的内容以文本形式直接显示到页面上。为了避免这样的情况发生，采用如下加 HTML 注释的方式(<! --注释-->)隐藏内容而不让它显示。

```
<head>
⋮
<style type="text/css">
<! --
hr {color: sienna}
p {margin- left: 20px}
body {background- image: url("images/back40.gif")}
-->
</style>
⋮
</head>
```

③ 导入外部样式表 。导入外部样式表是指在内部样式表的<style>里导入一个外部样式表，导入时用@import，见下面这个实例。

```
<head>
⋮
<style type="text/css">
<!--
@import "mystyle.css"
其他样式表的声明
-->
</style>
⋮
</head>
```

该实例中，@import "mystyle. css"表示导入 mystyle. css 样式表，注意使用外部样式表的路径，方法和链入样式表的方法很相似，但导入外部样式表输入方式更有优势。实质上它相当于存在内部样式表中。值得注意的是，导入外部样式表必须在样式表的开始部分，并在其他内部样式表上面。

④ 内嵌样式。内嵌样式是混合在 HTML 标记里使用的，用这种方法可以很简单地对某个元素单独定义样式。内嵌样式的使用是直接在 HTML 标记里加入 style 参数。而 style 参数的内容就是 CSS 的属性和值，如下例：

```
<p style="color: sienna;margin- left: 20px;">
这是一个段落
</p>
<!--这个段落颜色为土黄,左边距为 20 像素-->
```

在 style 参数后面的引号里的内容相当于在样式表大括号里的内容。注意：style 参数可以应用于任意 BODY 内的元素(包括 BODY 本身)，除了 BASEFONT、PARAM 和 SCRIPT。

特别要注意的是：如果在同一个选择器上使用几个不同的样式表时，这个属性值将会叠加几个样式表，遇到冲突的地方会以最后定义的为准。例如，首先链接到一个外部样式

表，其中定义了 h3 选择符的 color、text-alig 和 font-size 属性，代码如下。

```
h3
{
    color: red;
    text-align: left;
    font-size: 8pt;
}
/* 标题 3 的文字颜色为红色；向左对齐；文字尺寸为 8 号字 */
```

然后在内部样式表里也定义了 h3 选择符的 text-align 和 font-size 属性。

```
h3
{
    text-align: right;
    font-size: 20pt;
}
/* 标题 3 文字向右对齐；尺寸为 20 号字 */
```

那么这个页面叠加后的样式为

```
color: red;
text-align: right;
font-size: 20pt;
/* 文字颜色为红色；向右对齐；尺寸为 20 号字 */
```

字体颜色从外部样式表里保留下来，而对齐方式和字体尺寸都有定义时，按照后定义的优先。

4. 样式表常用属性

因篇幅关系，下面只列出样式表中常用的属性，见表 2-2。开发过程中可查阅 W3School。网址为 http://www.w3school.com.cn/css/index.asp。

表 2-2 样式表常用属性列表

属性名称	属性含义	属性值
字体属性(Font)		
font-family	使用什么字体	所有的字体
font-style	字体是否斜体	normal、italic、oblique
font-variant	是否用小体大写	normal、small-caps
font-weight	字体的粗细	normal、bold、bolder、lithter 等
font-size	字体的大小	absolute-size、relative-size、length、percentage 等
颜色和背景属性		
color	定义前景色	颜色
background-color	定义背景色	颜色

续表

属 性 名 称	属性含义	属　性　值
background-image	定义背景图案	路径
background-repeat	重复方式	repeat-x、repeat-y、no-repeat
background-attachment	设置滚动	scroll、fixed
background-position	初始位置	percentage、length、top、left、right、bottom 等
文本属性		
word-spacing	单词之间的间距	normal
letter-spacing	字母之间的间距	同上
text-decoration	文字的装饰样式	none|underline|overline|line-through|blink
vertical-align	垂直方向的位置	baseline| sub | super | top | text-top | middle | bottom | text-bottom|
text-transform	文本转换	capitalize|uppercase|lowercase|none
text-align	对齐方式	left|right|center|justify | normal| | |
text-indent	首行的缩进方式	
line-height	文本的行高	
边距属性		
margin-top	顶端边距	length|percentage|auto
margin-right	右侧边距	同上
margin-bottom	底端边距	同上
margin-left	左侧边距	同上
填充距属性		
padding-top	顶端填充距	length|percentage
padding-right	右侧填充距	同上
padding-bottom	底端填充距	同上
padding-left	左侧填充距	同上
边框属性		
border-top-width	顶端边框宽度	thin|medium|thick|length
border-right-width	右侧边框宽度	同上
border-bottom-width	底端边框宽度	同上
border-left-width	左侧边框宽度	同上
border-width	一次定义宽度	同上
border-color	设置边框颜色	color
border-style	设置边框样式	none|dotted|dash|solid 等
border-top	一次定义顶端	border-top-width|color 等
border-right	一次定义右侧	同上
border-bottom	一次定义底端	同上
border-left	一次定义左侧	同上
分级属性		
display	定义是否显示	block、inline、list-item、none

续表

属 性 名 称	属性含义	属 性 值
white-space	怎样处理空白	normal、pre、nowrap
list-style-type	加项目编号	disc、circle、square 等
list-style-image	加图案	none
list-style-position	第二行起始位置	inside、outside
list-style	一次定义列表	||

示例 4：使用 CSS 和 DIV 设计一个 Web 页面的布局效果图(见图 2-13)。

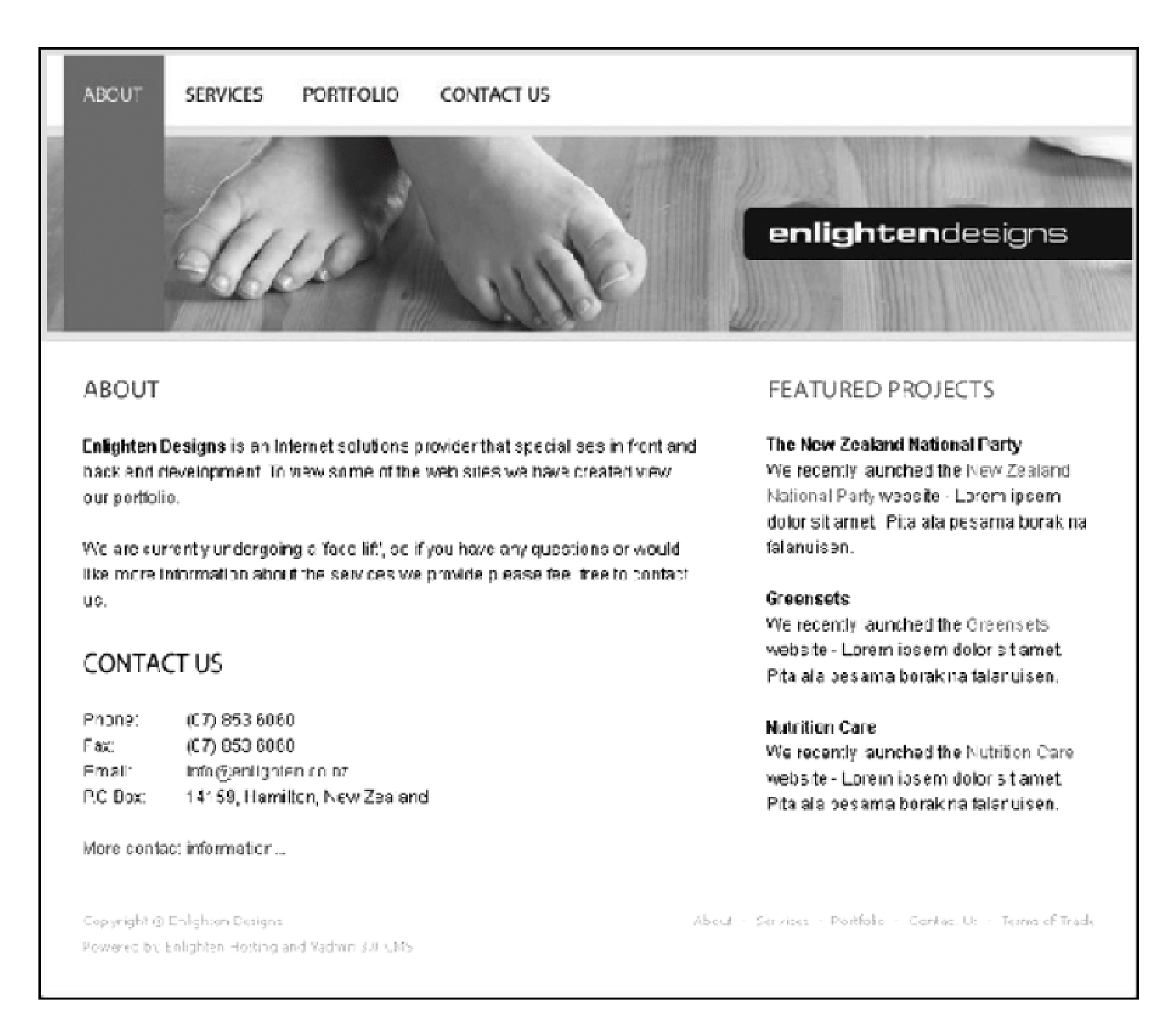


图 2-13　Web 页面效果图

① 规划网页。通过对图 2-13 分析，页面基本布局如图 2-14 所示。

- MAIN NAVIGATION：导航条，具有按钮特效。Width 为 760px，Height 为 50px。
- HEADER：网站头部图标，包含网站的 Logo 和站名。Width 为 760px，Height 为 150px。
- CONTENT：网站的主要内容。Width 为 480px，Height 会根据内容变化。
- SIDE BAR：边框，一些附加信息。Width 为 280px，Height 会根据内容变化。
- FOOTER：网站底栏，包含版权信息等。Width 为 760px，Height 为 66px。

② 新建 HTML 页。右击存放网页的文件夹，单击“添加新项”菜单项。在“名称”文本框中输入 Design. html，单击“HTML 页”列表项，再单击“添加”按钮。

③ 创建网站的大框，即建立一个宽为 760px 的盒子，它将包含网站的所有元素。在 Design. html 文件的<body>和</body>之间输入以下代码。

```
<div id="page-container">
    Hello world.
```

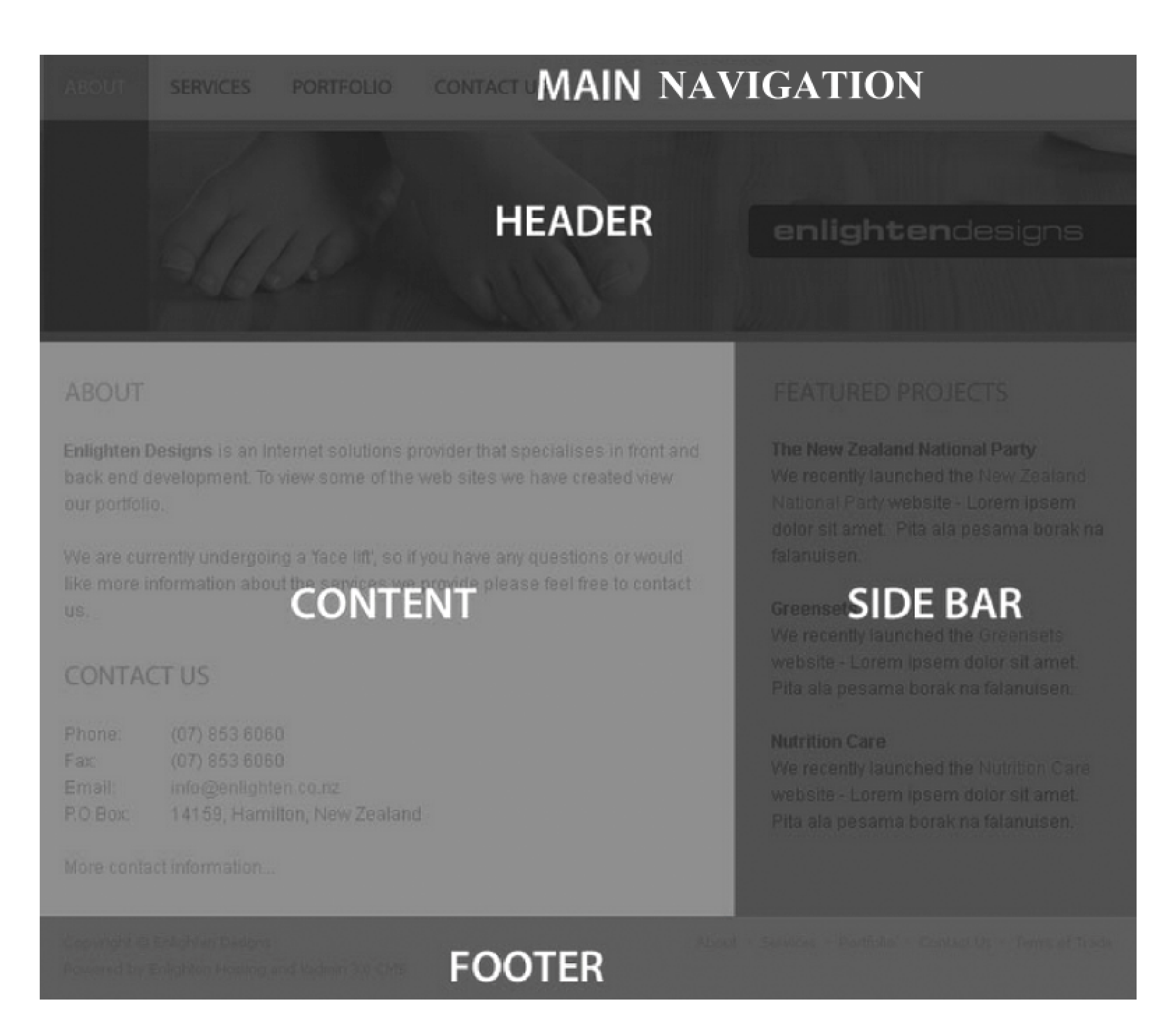


图 2-14　页面基本布局

```
</div>
```

创建 CSS 文件,命名为 design. css。在 design. css 中输入以下代码。

```
#page-container {
    width: 760px;
    margin: auto;
    background: red;
}
```

然后在 Design. html 中输入 link 指令,链接到 design. css 样式表。

```
<head>
    <title></title>
    <link href="design.css" rel="stylesheet" type="text/css" />
</head>
```

现在可以看到盒子和浏览器的顶端有 8px 宽的空隙。这是由于浏览器默认的填充和边界造成的。消除这个空隙,就需要在 design. css 文件中输入以下代码。

```
html, body {
    margin: 0;
    padding: 0;
}
```

④ 将页面划分为 5 个 DIV,实现页面基本布局的控制。在 Design. html 中输入加粗代码。

```
<div id="page-container">
    <div id="main-nav">Main Nav</div>
    <div id="header">Header</div>
```

```
        <div id="sidebar-a">Sidebar A</div>
        <div id="content">Content</div>
        <div id="footer">Footer</div>
</div>
```

为了将五个部分区分开来，将这五个部分用不同的背景颜色标示出来，在 design.css 文件中输入以下代码。

```
#main-nav {
     background: red;
     height: 50px;
}
#header {
     background: blue;
     height: 150px;
}
#sidebar-a {
     background: darkgreen;
}
#content {
     background: green;
}
#footer {
     background: orange;
     height: 66px;
}
```

至此，效果图如图 2-15 所示。



图 2-15　效果图 1

⑤ 网页布局与 DIV 浮动等。

首先让边框浮动到主要内容的右边，在 design.css 文件中输入以下代码。

```
#sidebar-a {
     float: right;
     width: 280px;
     background: darkgreen;
}
```

到这个时候网页效果如图 2-16 所示。在 content DIV 中输入任意显示内容。从图 2-17中看到，content DIV 盒子占据了整个 page-container 的宽度，需要将 # content 的右边界设为 280px，以使其不和边框发生冲突。在 design. css 中输入下列代码后，在 side bar DIV 中输入任意的显示内容(见图 2-18)。

```
#content {
    margin-right: 280px;
    background: green;
}
```


图 2-16　效果图 2

图 2-17　效果图 3

图 2-18 并不是设计所期望的结果，网站的底框跑到边框的下边去了。这是由于将边框向右进行了浮动，由于是浮动，所以可以理解为它位于整个盒子之上的另一层。因此，底框和内容盒子对齐了。进一步在 Design. css 文件中输入以下代码。

Main Nav

Header

Enlighten Design was established 2008 and is headed up by lighting consultant Rory Marr, who has worked in the lighting industry throughout Europe and the UK for the past 20 years. Great ideas need practical skills to make them a reality and with a collective ethos, Enlighten are now a team of expert associates covering the UK and sharing the same philosophy. Our consultants believe that comprehensible lighting advice, attention to detail and beautiful design, within brief and budget, is what makes our clients happy. We come from a variety of backgrounds, including product design, luminaire manufacturing, technical design, electrical engineering, wholesaling and the visual arts. Enlighten' s team are creative lighting designers sharing the passion, skills and experience to make a dramatic improvement to your property or development. Dealing with a wide range of lighting requirements throughout the UK, we bring the creative vision, detailed, technical and practical knowledge, to help you overcome any challenge and deliver fantastic results.

We come from a variety of backgrounds, including product design, luminaire manufacturing, technical design, electrical engineering, wholesaling and the visual arts. Enlighten' s team are creative lighting designers sharing the passion, skills and experience to make a dramatic improvement to your property or development. You used experts to design the building, now use experts to design the lighting. Whether you want to improve the look and feel of a room, reduce your maintenance costs and energy consumption or control the impact of your property's light upon the environment, we can deal with all your lighting requirements. Our consultancy service is independent, providing the best solutions to suit you. For advice on how to make the most of your project, avoid problems and resolve visual, technical and sustainability issues, talk to our

Footer

图 2-18　效果图 4

```
#footer {
    clear: both;
    background: orange;
    height: 66px;
}
```

⑥ 网页主要框架之外的附加结构的布局与表现。

除网页主要框架之外的附加结构的表现(Layout)包括以下内容：标题(heading)，包括网站名、内容标题、页脚信息、版权、认证、副导航条。现在来加入标题。先回到 HTML 的代码，<h1>～<h6>是常用的 HTML 标题代码。比如一般为：<h1>网站名</h1>，<h2>网站副标题</h2>，<h3>内容主标题</h3>。向 HTML 文件的 Header 层(Div)输入以下代码。

```
<div id="header">
    <h1>Enlighten Designs</h1>
</div>
```

刷新一下页面，就可以看到巨大的标题和标题周围的空白，这是由<h1>标签的默认大小和边距(margin)造成的，要消除这些空白，在 CSS 文件中输入以下代码。

```
h1 {
    margin: 0;
    padding: 0;
}
```

接下来继续页脚的制作。页脚包括两部分：左边的版权、认证和右边的副导航条。先要让副导航条向右浮动，就像之前处理 Side bar 和 Content 一样，需要在页面中加入一个新的层(div)。

```
<div id="footer">
    <div id="altnav">
        <a href="http://css.jorux.com/wp-admin/post.php#">About</a> -
        <a href="http://css.jorux.com/wp-admin/post.php#">Services</a> -
        <a href="http://css.jorux.com/wp-admin/post.php#">Portfolio</a> -
        <a href="http://css.jorux.com/wp-admin/post.php#">Contact Us</a> -
        <a href="http://css.jorux.com/wp-admin/post.php#">Terms of Trade</a>
    </div>
    Copyright © Enlighten Designs
</div>
```

⑦ 页面内的基本文本的样式(CSS)设置。

如果不喜欢太花哨的背景，现在可以去掉，只保留导航条的红色背景。在 Design. css 文件中输入以下代码。

```
body {
    font-family: Arial, Helvetica, Verdana, Sans-serif;
    font-size: 12px;
    color: #666666;
    background: #ffffff;
}
```

一般把 body 标签放在 CSS 文件的顶端。font-family 内的顺序决定了字体显示的优先级，比如，如果计算机没有 Arial 字体，浏览器就会指向 Helvetica 字体，以此类推；color 指字体颜色；background 指背景颜色。现在的网页效果如图 2-19 所示。

⑧ 网站头部图标与 Logo 部分的设计。

接着给＃header 层添加背景图案。在 CSS 文件中输入以下内容。

```
#header {
    height: 150px;
    background: #db6d16
    url(../images/headers/about.jpg);
}
```

接着替换掉“<h1></h1>”标签里的 Enlighten Designs。

```
<div id="header">
<h1><img src="http://img.jcwcn.com/attachment/portal"BORDER-RIGHT: #4b8be3 1px solid; BORDER-TOP: #
4b8be3 1px solid; BORDER-LEFT: #4b8be3 1px solid; BORDER-BOTTOM: #4b8be3 1px solid" height="60" width="
550" bgcolor="#dfeeff">
#header h1 {
    margin: 0;
```



图 2-19　效果图 5

```
    padding: 0;
    float: right;
    margin-top: 57px;
    padding-right: 31px;
}
```

这样使存在于<h1>层的图片向右浮动，并且上边距(margin-top)为 57px，右间隙(padding-right)为 31px。

2.3.6　使用 JavaScript 客户端编程

JavaScript 是一种基于对象和事件驱动，具有安全性能的脚本语言。JavaScript 被嵌入到 HTML 文件中，不需要经过 Web 服务器就可以对用户操作做出响应，使网页更好地与用户交互；在充分利用客户端计算机性能的同时，可以适当减小服务器端的压力，并减少用户的等待时间。与在网页中插入 CSS 的方式相似，使用<script>标签在网页中插入 JavaScript 代码。使用下面的代码可以在网页中插入 JavaScript。

```
<script type="text/JavaScript" language="javascript">
</script>
```

JavaScript 脚本可以放在网页的 head 部分或者 body 部分，也可以放在外部的 JavaScript 文件中，执行效果有区别。

(1) 放在 body 部分的 JavaScript 脚本在网页读取到该语句的时候就会执行。例如：

```
<html>
    <body>
```

```
        <script type="text/JavaScript">
            document.write("我是菜鸟我怕谁");
        <script>
    <body>
</html>
```

(2) 在 head 部分的脚本在被调用的时候才会执行,例如:

```
<html>
    <head>
        <script type="text/JavaScript">
        ⋮
        </script>
    </head>
<html>
```

通常是在<script>...</script>部分定义函数,通过调用函数来执行 head 部分的脚本。

(3) 也可以像添加外部 CSS 一样添加外部 JavaScript 脚本文件,其扩展名通常为.js。例如:

```
<html>
    <head>
        <script src="scripts.js"></script>
    </head>
    <body>
    </body>
</html>
```

如果很多网页都需要包含一段相同的代码,那么将这些代码写入一个外部 JavaScript 文件中是最好的方法。此后,任何一个需要该功能的网页,只需要引入这个 js 文件就可以了。

注意: 脚本文件里头不能再含有<script>标签。但是放在 head 部分的函数是一个例外,它只有被调用时才会执行。

1. JavaScript 变量

在代数中,通常会遇到下面的基础问题,如果 a 的值为 5,b 的值为 6,那么 a 与 b 的和是多少? 在这个问题中,我们就可以把 a 和 b 看作变量,再设置一个变量 c 来保存 a 与 b 的和。

那么,上面的这个问题就可以用如下的 JavaScript 代码表示:

```
<script type="text/javascript">
    //计算 a+b的和
    a=5;//给变量 a赋值
    b=5;//给变量 b赋值
    c=a+b;//c为 a+b的和
    document.write(c);//输出 c的值
</script>
```

输出结果为 10。

在上面的例子中,用到了三个变量 a、b、c,这些都是变量的名字。在 JavaScript 中,需要用变量名来访问这个变量。变量名有如下规定:

- 变量名区分大小写，A 与 a 是两个不同的变量。
- 变量名必须以字母或者下划线开头。

也可以用 var 声明变量，例如：

```
<script type="text/javascript">
    var a;    //声明一个变量 a
    a=5;      //给变量赋值
</script>
```

JavaScript 允许不声明变量直接赋值。不过先声明变量是一个良好的编程习惯。在 JavaScript 中，变量是无所不能的容器，可以把任何东西存储在变量里，例如：

```
var quanNeng1=123;          //数字
var quanNeng2="一二三"  //字符串
```

其中，quanNeng2 这个变量存储了一个字符串，字符串需要用一对引号括起来。变量还可以存储更多的东西，例如数组、对象、布尔值等。

2. JavaScript 操作符

操作符是用于在 JavaScript 中指定一定动作的符号，其中算术操作符主要用来完成类似加减乘除的工作。看下面这段 JavaScript 代码：

```
c=a+b;
```

其中的“＝”和“＋”都是操作符。JavaScript 中还有很多这样的操作符，例如，加减乘除是 JavaScript 中比较基本的几个操作符，它们的意义与在数学中没有什么差别。JavaScript 中最常见的操作符是赋值操作符“＝”，上一节我们已经强调过，它不是“等于”。

（1）操作符的优先级。

在数学中，“a＋b＊c”这个式子中，乘法将先于加法运算。同样，在 JavaScript 中，这个式子会按相同的顺序执行，被称为“优先级”，即“＊”的优先级高于“＋”。

与数学中一样，改变运算顺序的方法是添加括号，JavaScript 中改变优先级的方法也是添加括号。例如：

```
(a+b) * c
```

（2）字符串的连接。

在 JavaScript 中，“＋”不只代表加法，同样也可以使用它来连接两个字符串，例如：

```
example="乌"+"龟";
```

在上面的例子中，example 将包含“乌龟”这个字符串。这是由于“＋”完成了“乌”和“龟”的连接，当然也可以把这种行为理解成字符串的加法。

（3）自加一、自减一操作符。

接着来看两个非常常用的运算符，自加一用“＋＋”；自减一用“--”。首先来看一个例子：

```
a=5;
a++;      //a 的值变为 6
a--       //a 的值又变回 5
```

上面的例子中,a++使得 a 的值在原来的基础上增加 1,a--则让 a 在现在的基础上再减去 1。所以,其实“a++”也可以写成:

```
a=a+1;
```

3. 复合语句

(1) if else 结构

if else 的意思和字面意思是一样的,就是“如果”、“否则”。再来看一个使用 if 的例子。

```
<script type="text/JavaScript">
    Var hobby="VbScript";
    if (hobby =="JavaScript")
    {
        document.write("有发展");
    }
</script>
```

接下来解释一下这段代码。首先是 if 后面紧跟着一个括号,括号里则是一个条件,确切地说是一个布尔值。当条件成立的时候,这个值是 True,“{}”里的语句将会得到执行;否则这个值是 False,“{ }”里的语句将被忽略。具体到该例子,因为 hobby 变量的值是 VbScript,所以不必处理;如果 hobby 变量的值是 JavaScript,则输出“有发展”。注意“==”,这个符号用来判断左右两边是否相等。如果选择的爱好不是 JavaScript,那么没有任何输出。如果希望程序能对这种情况做出反应,可以用 else。看下面的代码。

```
<script type="text/JavaScript">
    var hobby="JavaScript"
        if (hobby =="JavaScript")
        {
            document.write("有发展");
        }
            else//如果爱好不是 JavaScript
        {
            document.write("没有评价...");
        }
</script>
```

上面的代码用到了 else,它会给 if 添加一种“否则”的状态。当 hobby 变量值不是 JavaScript 的时候,它会输出“没有评价”。

如果想做更多的判断,可以用 if 的嵌套,看下面的代码。

```
<script type="text/JavaScript">
    var hobby="JavaScript"
    if(hobby=="JavaScript")
    {
        document.write("有发展");
    }
    else if(hobby=="football")//如果爱好是足球
    //注意,这个 if是嵌套在上一个 if else中的 else部分
    {
```

```
        document.write("我 X");
    }
    else//既不是 JavaScript 又不是足球
    {
        document.write("没有评价……");
    }
</script>
```

第二个 if 只有在第一个 if 的条件不成立的时候才有机会执行。

(2) for 循环

所谓循环,就是重复执行一段代码。for 语句结构如下:

```
for(初始条件;判断条件;循环后动作)
{
    循环代码
}
```

先来看一个简单的例子。有 10 个菜鸟报数,“菜鸟 1 号、菜鸟 2 号……”。有了 for 循环,很少的代码就可以实现 10 个菜鸟的报数。

```
<script type="text/JavaScript">
    var i=1;
    for(i=1;i<=10;i++)
    {
        document.write("菜鸟"+i+"号<br />");
    }
</script>
```

输出结果如下:

```
菜鸟 1号
菜鸟 2号
菜鸟 3号
菜鸟 4号
菜鸟 5号
菜鸟 6号
菜鸟 7号
菜鸟 8号
菜鸟 9号
菜鸟 10号
```

在这个例子中,循环恰好执行了 10 次,那么“for(i=1;i<=10;i++)”一句中的 10 是不是 10 次的意思呢?下面就来看看 for 循环的工作机制。

首先“i=1”叫作初始条件,也就是说从哪里开始,该例子从 i=1 开始。出现在第一个分号后面的“i<=10”表示判断条件,每次循环都会先判断这个条件是否满足,如果满足则继续循环,否则停止循环,继续执行 for 循环后面的代码。那么设定了 i=0,那么,是不是 i 永远都小于等于 10 呢?来看第三个部分。最后的“i++”表示让 i 在自身的基础上加 1,这是每次循环后的动作。也就是说,每次循环结束,i 都会比原来大 1,执行若干次循环之后,i≤10 的条件就不满足了,这时循环结束。for 循环后面的代码将被执行。

（3）while 循环

while 循环重复执行一段代码，直到某个条件不再满足。

① while 循环的结构

```
while(判断条件)
{
    循环代码
}
```

其实 while 循环和 for 循环的作用都是重复执行代码，例如下面这段代码，与上一节 for 循环的输出结果完全没有区别。

```
<html>
  <body>
    <script type="text/JavaScript">
    var i=0;
      while (i<=10)
      {
        document.write("菜鸟"+i+"号");
        document.write("<br />");
        i=i+1;
      }
    </script>
  </body>
</html>
```

以上代码看起来好像比 for 循环少了条件，即只有一个判断条件。其实这个循环也是有初始条件的，只不过已经定义好了，即“var i=0;”。变量 i 的增大，则是放到了循环体里面，其实这个过程和 for 循环没有什么区别，也是变量 i 不断变大，直到判断条件不满足，则循环结束。

② do while 循环的结构

do wile 结构的基本原理和 while 结构是基本相同的，但是它保证循环体至少被执行一次。因为它是先执行代码，后判断条件。代码如下。

```
<script type="text/JavaScript">
    i=0;
    do
    {
        document.write("The number is "+i);
        document.write("<br />");
        i++;
    }
    while(i <=5)
</script>
```

（4）break 和 continue

break 可以跳出循环，continue 跳过本次循环。break 语句可以让循环中途停止，直接执行后面的代码。格式如下。

```
while (i<10)
```

```
{
    if(特殊情况)
    break;
    循环代码
}
```

当特殊情况发生的时候,循环就会立即结束。看看下面的例子,菜鸟 7 号到菜鸟 10 号在寝室打游戏。

```
<html>
    <body>
        <script type="text/JavaScript">
        var i=0;
        for (i=0;i<=10;i++)
        {
            if(i==6)
            {
                break;//如果 i 是 6 的话就退出循环
            }
            document.write("菜鸟"+i+"号<br/>");
        }
        </script>
    </body>
</html>
```

当 i=7 的时候循环就会结束,不会输出后面循环的内容。

continue 的作用是仅仅跳过本次循环,而整个循环体继续执行。它的格式如下。

```
while(判断条件)
{
    if(特殊情况)
        continue;
    循环代码
}
```

上面的循环中,当特殊情况发生的时候,本次循环将被跳过,而后续的循环则不会受到影响,来看看下面的例子:菜鸟 6 号外出学习 JavaScript 去了。

```
<html>
    <body>
        <script type="text/JavaScript">
        var i=0
        for(i=0;i<=10;i++)
        {
            if(i==3)
            {
                continue;
            }
            document.write("The number is "+i);
            document.write("<br />");
        }
```

```
        </script>
    </body>
</html>
```

上面的代码中，i=6 的那次循环将被跳过。

(5) JavaScript 函数

通常情况下，函数是完成特定功能的一段代码。把一段完成特定功能的代码块放到一个函数里，以后就可以调用这个函数，省去重复输入大量代码的麻烦。一个函数的作用就是完成一项特定的任务。如果没有函数时，完成一项任务可能需要 5 行、10 行甚至更多的代码。每次需要完成这个任务的时候都重写一遍代码显然不是一个好主意。这时可以编写一个函数来完成这个任务，以后只要调用这个函数即可。定义函数格式如下。

```
function 函数名 ()
{
    函数代码;
}
```

函数由关键字 function 定义，把"函数名"替换为想要的名字，把"函数代码"替换为完成特定功能的代码。了解了如何定义函数，就可以编写一个实现两数相加的简单函数。首先给函数起一个有意义的名字 add2，它的代码如下。

```
function add2(){
    sum=1+1;
    alert(sum);
}
```

函数定义好以后，可以通过很多种方法调用，这里使用最简单的函数调用方式——按钮的单击事件。试着单击下面的按钮来调用已经定义好的函数。

```
<html>
<head>
<script language=javascript>
    function add2(){
    sum=1+1;
    alert(sum);
}
</SCRIPT>
</head>
<body>
<form>
<input type="button" value="click it" ONCLICK=" add2 ()">
</form>
</body>
</html>
```

以上代码通过 button 按钮的单击事件 onclick 调用 add2()函数。上述的 add2()函数不能实现任意指定的两数相加。函数的定义也可以用下面的格式：

```
function 函数名(参数 1,参数 2,参数 3){
    函数代码...
}
```

按照这个格式，函数可以改写成：

```
function add2(x,y){
    sum=x+y;
    alert(sum);
}
```

x 和 y 是 add2 函数的两个参数，调用函数的时候，通过这两个参数把两个加数传递给了函数。例如，add2(3,4)会求 3＋4 的和，add2(56,65)则会求出 56 和 65 的和。alert(sum)一行改成下面的代码较易理解：

```
return sum;
```

return 后面的值叫作返回值。使用下面的语句调用函数就可以将这个返回值存储在变量中。

```
result=add2(3,4);
```

该语句执行后，result 变量中的值为 7。需要说明的是，在该示例中参数和返回值都是数字，其实它们也可以是字符串或其他类型。

(6) JavaScript 事件

函数定义之后，默认是不会执行的，这时候就需要一些事件来触发这个函数并使其被执行。JavaScript 有很多事件，例如鼠标的单击、移动，网页的载入和关闭。先来定义一个函数，再看几个事件的实例。示例函数如下。

```
<script type="text/JavaScript">
function displaymessage()
{
    alert("我是菜鸟我怕谁!");
}
</script>
```

函数的事件很简单，只是显示一条消息。使用单击事件，需要给元素设置 onclick 属性。示例代码如下。

```
< button value="单击“提交”按钮" onclick="displaymessage()">onclick 调用函数</button>
```

由于设置了 onclick="displaymessage()"，因此单击按钮则会调用函数。使用鼠标经过事件调用函数的代码如下。

```
<button value="单击“提交”按钮" onmouseover="displaymessage()">鼠标指针滑过调用函数</button>
```

当鼠标经过按钮时，触发 onmouseover 事件，将会调用函数 displaymessage()。使用鼠标移出事件调用函数的代码如下。

```
<button value="单击“提交”按钮" onmouseout="displaymessage()">鼠标指针移出调用函数</button>
```

把鼠标移动到这个按钮里面，当再移动出去时，触发 onmouseout 事件，将会调用函数 displaymessage()。限于篇幅，本书中不再一一列举事件列表，具体可参看 W3School JavaScript HTML DOM 事件，地址为：http://www.w3school.com.cn/js/js_htmldom_events.asp。

示例 5：使用 JavaScript 和 CSS 制作下拉菜单，如图 2-20 所示。



图 2-20　下拉菜单示例

① 新建 index.html 文件和 default.css 文件。在 index.html 中输入以下代码（也可使用 Dreamweaver 制作界面），效果如图 2-21 所示。

- 主页
  HTML DropDown DHTML DropDown menu JavaScript DropDown DropDown Menu CSS DropDown
- 下载
  ASP Dropdown Pulldown menu AJAX dropdown DIV dropdown
- 订单
  Visa Credit Card Paypal
- 帮助
  Download Help File Read online
- 联系我们
  E-mail Submit Request Form Call Center

图 2-21　下拉列表效果图 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.
dtd">
<html>
<head>
<title></title>
    <meta name="keywords" content="">
    <meta name="description" content="">
</head>
<body>
<ul id="sddm">
    <li><a href="#" >主页</a>
        <div id="m1" ">
        <a href="#">HTML DropDown</a>
        <a href="#">DHTML DropDown menu</a>
        <a href="#">JavaScript DropDown</a>
        <a href="#">DropDown Menu</a>
        <a href="#">CSS DropDown</a>
        </div>
    </li>
    <li><a href="#" >下载</a>
        <div id="m2" >
        <a href="#">ASP Dropdown</a>
        <a href="#">Pulldown menu</a>
        <a href="#">AJAX dropdown</a>
        <a href="#">DIV dropdown</a>
        </div>
    </li>
    <li><a href="#" ">订单</a>
        <div id="m3" ">
```

```
            <a href="#">Visa Credit Card</a>
            <a href="#">Paypal</a>
            </div>
        </li>
        <li><a href="#">帮助</a>
            <div id="m4" ">
            <a href="#">Download Help File</a>
            <a href="#">Read online</a>
            </div>
        </li>
        <li><a href="#" ">联系我们</a>
            <div id="m5" ">
            <a href="#">E-mail</a>
            <a href="#">Submit Request Form</a>
            <a href="#">Call Center</a>
            </div>
        </li>
</ul>
<div style="clear:both"></div>
</body>
</html>
```

② 在 default.css 文件中输入下列代码。

```
#sddm
{   margin: 0;
    padding: 0;
    z-index: 30}

#sddm li
{   margin: 0;
    padding: 0;
    list-style: none;
    float: left;
    font: bold 11px arial}

#sddm li a
{   display: block;
    margin: 0 1px 0 0;
    padding: 4px 10px;
    width: 60px;
    background: #5970B2;
    color: #FFF;
    text-align: center;
    text-decoration: none}

#sddm li a:hover
{   background: #49A3FF}

#sddm div
{   position: absolute;
```

```
    visibility: hidden;
    margin: 0;
    padding: 0;
    background: #EAEBD8;
    border: 1px solid #5970B2}

#sddm div a
{   position: relative;
    display: block;
    margin: 0;
    padding: 5px 10px;
    width: auto;
    white-space: nowrap;
    text-align: left;
    text-decoration: none;
    background: #EAEBD8;
    color: #2875DE;
    font: 11px arial}

#sddm div a:hover
{   background: #49A3FF;
    color: #FFF}
```

接着在 index. html 中输入<link rel="stylesheet" type="text/css" href="css/default. css">,效果如图 2-22 所示。

主页	下载	订单	帮助	联系我们

图 2-22　下拉列表效果图 2

③ 在 index. html 文件的<head></head>部分输入以下代码。

```
<script type="text/javascript">
<!--
var timeout         = 500;
var closetimer      = 0;
var ddmenuitem      = 0;

// open hidden layer
function mopen(id)
{
    // cancel close timer
    mcancelclosetime();

    // close old layer
    if(ddmenuitem) ddmenuitem.style.visibility='hidden';

    // get new layer and show it
    ddmenuitem=document.getElementById(id);
    ddmenuitem.style.visibility='visible';
```

```
}
// close showed layer
function mclose()
{
    if(ddmenuitem) ddmenuitem.style.visibility='hidden';
}

// go close timer
function mclosetime()
{
    closetimer=window.setTimeout(mclose, timeout);
}

// cancel close timer
function mcancelclosetime()
{
    if(closetimer)
    {
        window.clearTimeout(closetimer);
        closetimer=null;
    }
}

// close layer when click-out
document.onclick=mclose;
// -->
</script>
```

这段 JS 代码中，定义了 mopen(id)、mclose()、mclosetime()和 mcancelclosetime()四个函数，分别用于打开层、关闭层、设置定时器和取消定时器。

④ 接下来为顶层菜单项和 div 添加 onmouseover 和 onmouseout 事件。如代码黑色加粗部分所示。最终效果如图 2-22 所示。

```
<ul id="sddm">
    <li><a href="#" onmouseover="mopen('m1')" onmouseout="mclosetime()">主页</a>
        <div id="m1 "onmouseover="mcancelclosetime()" onmouseout="mclosetime()">
        <a href="#">HTML DropDown</a>
        <a href="#">DHTML DropDown menu</a>
        <a href="#">JavaScript DropDown</a>
        <a href="#">DropDown Menu</a>
        <a href="#">CSS DropDown</a>
        </div>
    </li>
    <li><a href="#"onmouseover="mopen('m2')" onmouseout="mclosetime()">
    下载</a>
        <div id="m2" onmouseover="mcancelclosetime()" onmouseout="mclosetime()">
        <a href="#">ASP Dropdown</a>
        <a href="#">Pulldown menu</a>
        <a href="#">AJAX dropdown</a>
        <a href="#">DIV dropdown</a>
```

```
            </div>
        </li>
        <li><a href="#" onmouseover="mopen('m3')" onmouseout="mclosetime()">
        订单</a>
            <div id="m3" onmouseover="mcancelclosetime()" onmouseout="mclosetime()">
            <a href="#">Visa Credit Card</a>
            <a href="#">Paypal</a>
            </div>
        </li>
        <li><a href="#" onmouseover="mopen('m4')" onmouseout="mclosetime()">
        帮助</a>
            <divid="m4" onmouseover="mcancelclosetime()" onmouseout="mclosetime()">
            <a href="#">Download Help File</a>
            <a href="#">Read online</a>
            </div>
        </li>
        <li><a href="#" onmouseover="mopen('m5')" onmouseout="mclosetime()">
        联系我们</a>
            <div id="m5" onmouseover="mcancelclosetime()" onmouseout="mclosetime()">
            <a href="#">E-mail</a>
            <a href="#">Submit Request Form</a>
            <a href="#">Call Center</a>
            </div>
        </li>
    </ul>
```

24 项 目 实 施

2.4.1 任务 1：建立 Smart On Line 电子商城站点，添加网站横幅广告

1. 任务目标

(1) 能熟练操作 VS2012 集成开发环境。

(2) 能熟练使用 VS2012 创建 Web 站点。

(3) 能运用 VS2012 添加和编辑 Web 页面。

(4) 能熟练设置控件属性。

2. 任务内容

(1) 创建 Smart On Line 站点。

(2) 添加和编辑首页页面。

(3) 在首页中添加横幅广告。

3. 任务实施步骤

该任务的实现具体分为创建 Smart On Line 站点、添加 Web 页面、在首页中添加横幅广告三个步骤。

(1) 创建 Smart On Line 站点

下面介绍创建 Smart On Line 站点详细步骤。首先单击 Visual Studio 2012 图标启动 VS2012，如图 2-23 所示。

接着选择“新建”→“网站”命令，出现“新建网站”对话框，如图 2-24 和图 2-25 所示。

在对话框中单击“ASP. NET 空网站”选项，并且输入所要保存的文件位置，示例中将其保存到 C 盘根目录下，并且将网站名更改为 SmartOnLine。单击“确定”按钮，创建新的空白网站结构，如图 2-26 所示。新创建的网站中只有 Web. config 文件。Web. config 文件是一个 XML 文本文件，用来储存 ASP. NET Web 应用程序的配置信息（如设置 ASP. NET Web 应用程序的身份验证方式），它可以出现在应用程序的每一个目录中。当通过. NET 新建一个 Web 应用程序后，默认情况下会在根目录自动创建一个默认的 Web. config 文件，所有的子目录都继承它的配置。如果想修改子目录的配置，可以在该子目录下新建一个 Web. config 文件。它提供除从父目录继承的配置信息以外的信息，也可以重写或修改父目录中定义的设置。在运行时对 Web. config 文件的修改不需要重启服务就可以生效（注：＜processModel＞部分例外）。当然 Web. config 文件是可以扩展的。可以自定义新配置参数并编写配置节处理程序以对它们进行处理。



图 2-23　启动 VS2012

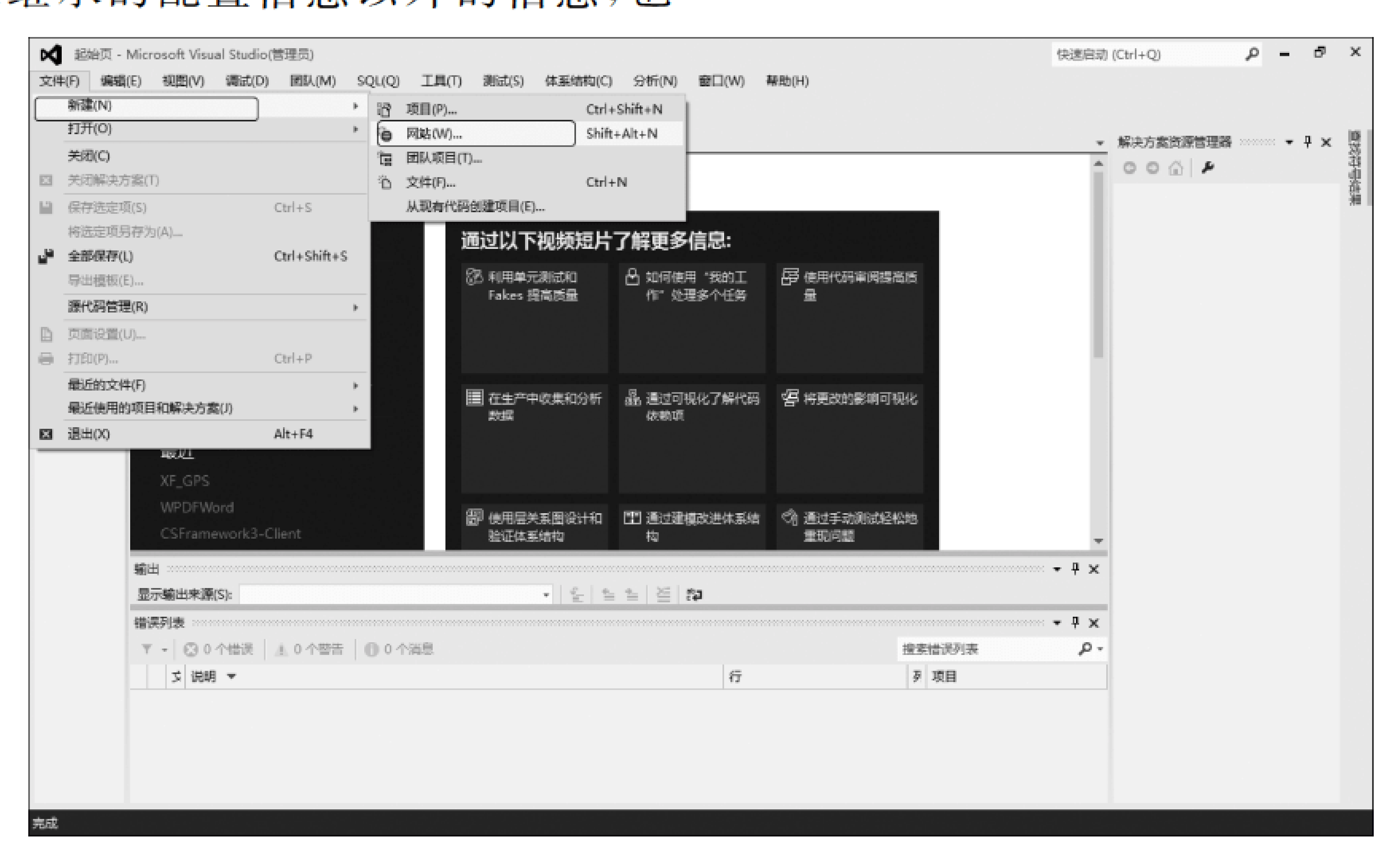


图 2-24　选择“新建”→“网站”命令

注意：刚创建的网站中尚没有任何的空白页面，需要添加新的 Web 页面。

（2）添加 Web 页面



图 2-25　“新建网站”对话框

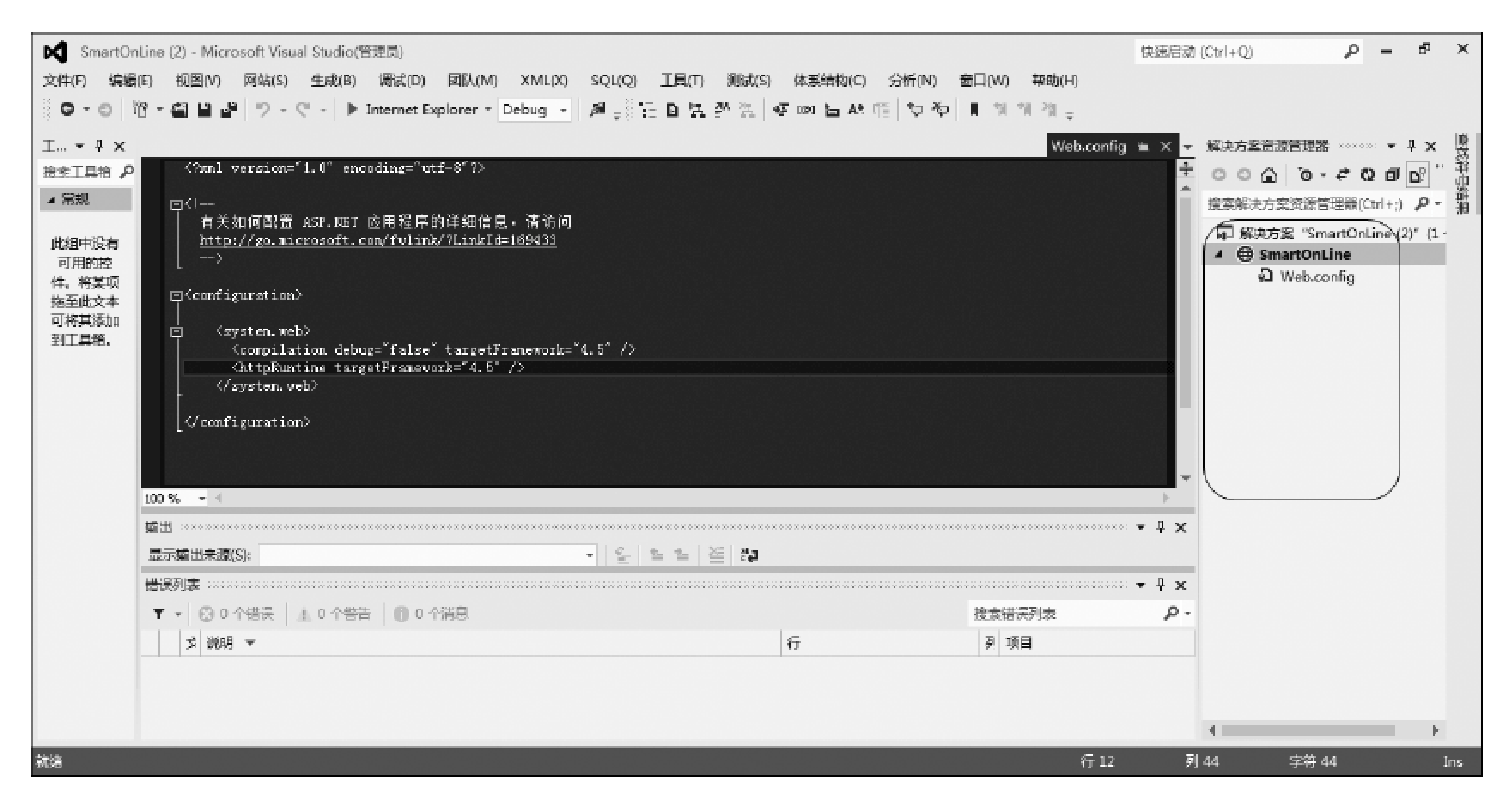


图 2-26　网站结构图

右击 SmartOnLine 项目文件夹，接着选择“添加新项”命令，如图 2-27 所示。接着出现“添加新项”对话框，如图 2-28 所示。

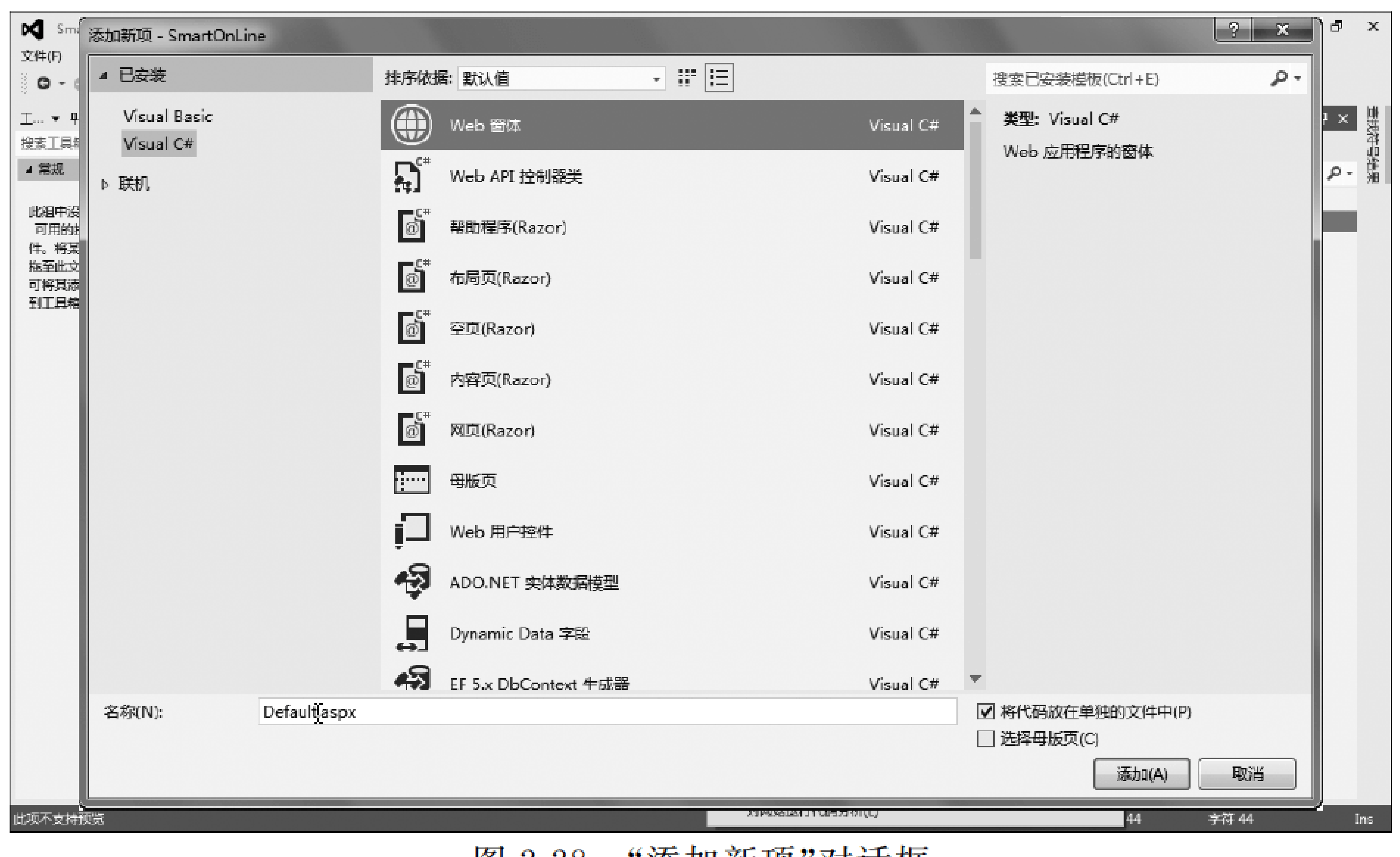


图 2-28　“添加新项”对话框

图 2-27　选择“添加新项”命令

在“添加新项”对话框中单击“Web 窗体”选项，更改相应的名称。这里将首页命名为 Default.aspx。单击“确定”按钮，出现如图 2-29 所示的空白 Web 页面。

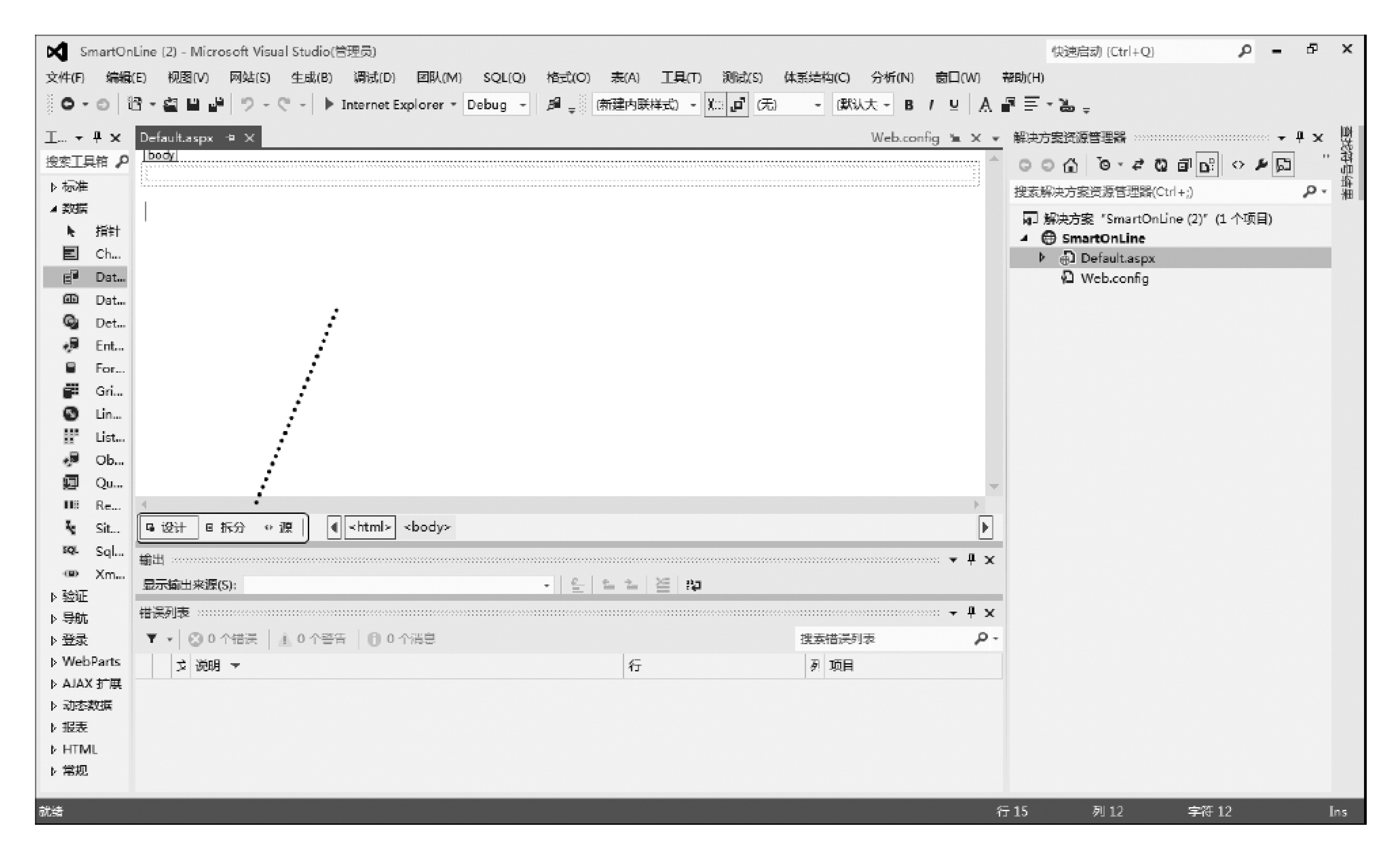


图 2-29　空白 Web 页面

单击“源”选项卡，切换到“源”视图方式(如图 2-30 所示)。

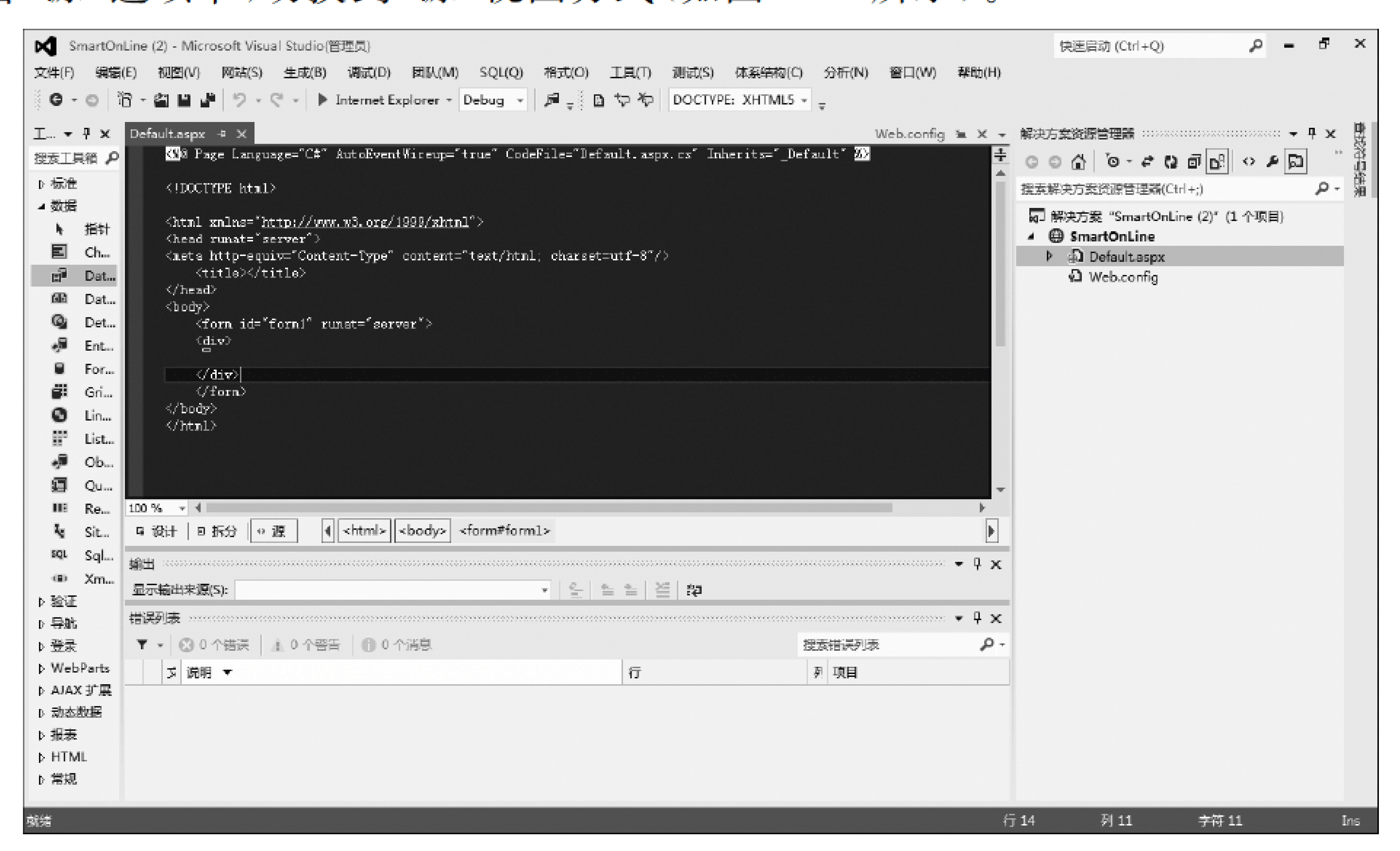


图 2-30　代码视图

在代码视图中修改标题，将 title 代码更改为“<title>欢迎您访问 Smart On Line </title>”。

接着单击 Debug 按钮，如图 2-31 所示。

在第一次启动运行时会弹出“未启用调试”提示信息，如图 2-32 所示。为便于调试，在测试阶段应单击“修改 web.config 文件以启用调试”选项，运行效果如图 2-33 所示。

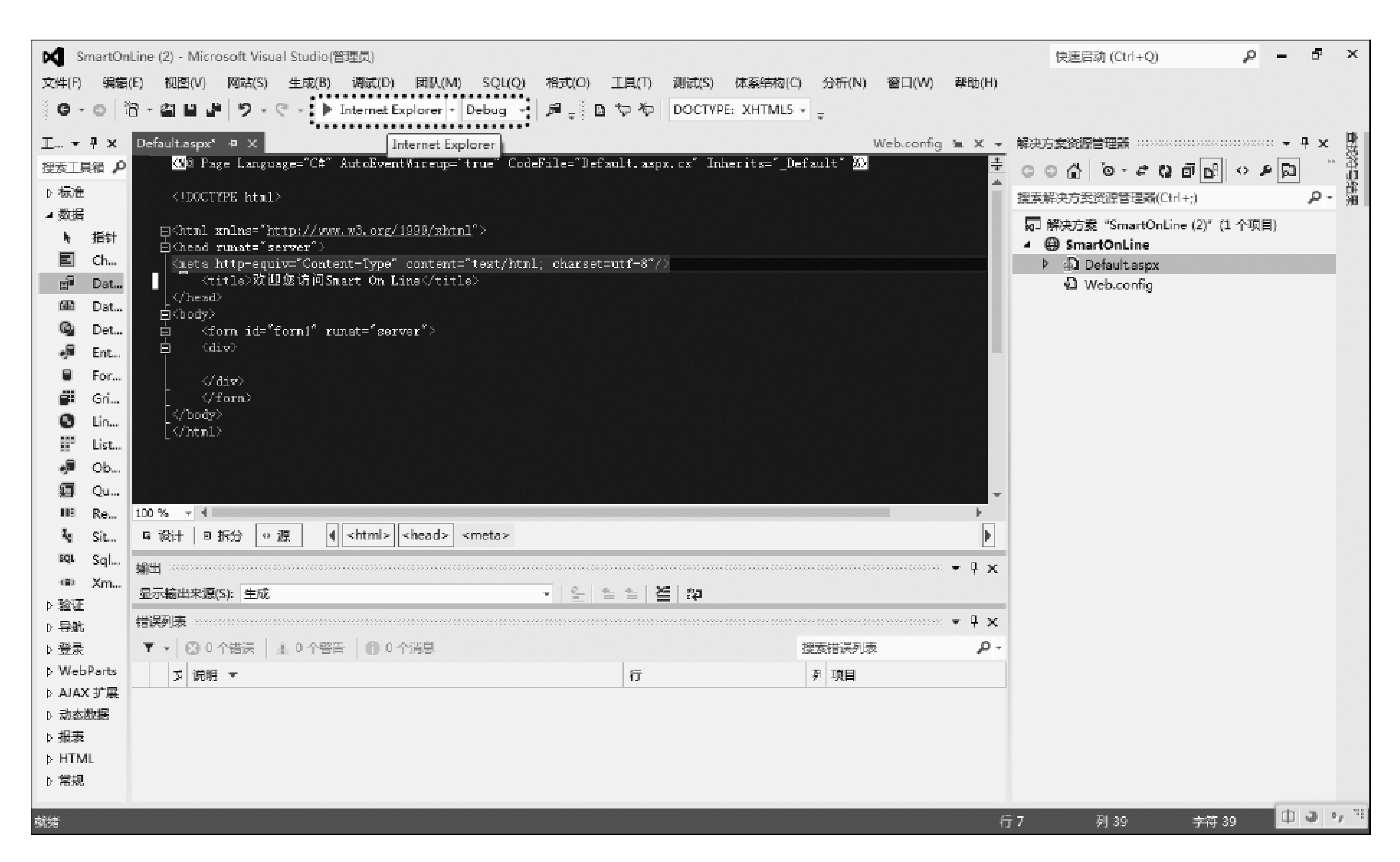


图 2-31　启用调试



图 2-32　“未启用调试”提示信息



图 2-33　修改标题后的网站运行效果

(3) 添加 SmartOnLine 首页横幅广告

① 便于清晰显示网站结构和控制访问权限，需要在站点下新建文件夹中单独存放图片。右击 SmartOnLine 项目根文件夹，弹出“添加”菜单，接着单击“新建文件夹”菜单项(见

图 2-34),将文件夹更名为 Image。

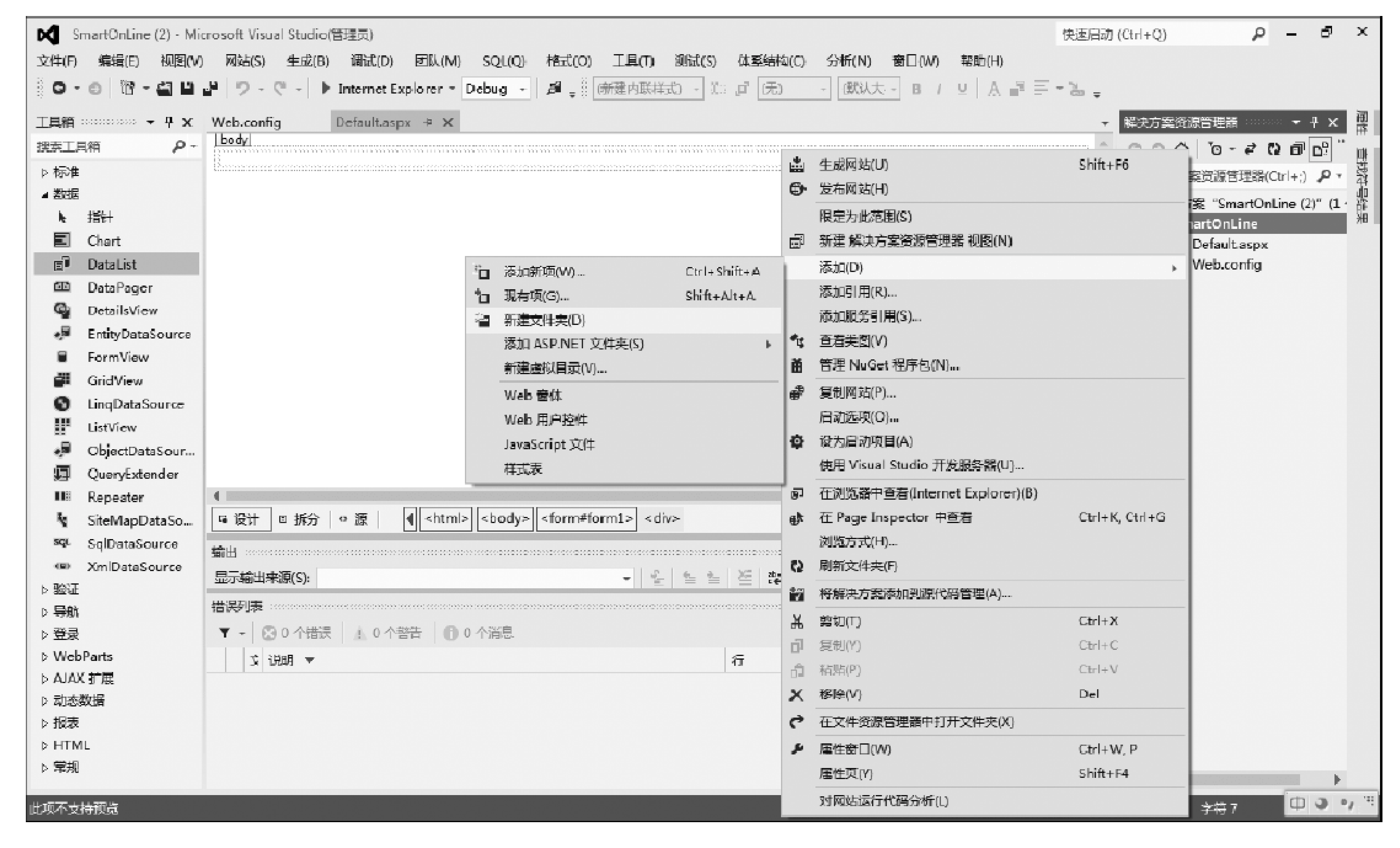


图 2-34 新建文件夹

② 将素材中的 Banner.png 复制到 Image 文件夹中。拖放 Image 控件到 Default.aspx 页面中,其 ID 为 Image1。

③ 右击 Image1 控件,接着单击"属性"菜单项显示属性页,按照表 2-3 所示修改控件属性。

表 2-3 更改 Image1 控件属性

属 性	值	属 性	值
ID	imgBanner	ImageUrl	~/Image/banner.png

④ 单击"运行"按钮,显示的效果图如图 2-35 所示。



图 2-35 任务 1 的最终效果图

2.4.2 任务 2：制作 Smart On Line 商城母版页

1. 任务目标

(1) 能使用 DIV 和 CSS 完成 Smart On Line 商城母版页的布局。

(2) 能熟练使用 VS2012 创建母版页。

2. 任务内容

(1) 能熟练使用 VS2012 创建母版页。

(2) 运用 DIV 和 CSS 设计 Smart On Line 商城母版页布局。

3. 任务实施步骤

制作 Smart On Line 商城母版页(见图 2-36)细化为两个小环节。第一个环节是创建母版页文件;第二个环节则是在母版页中使用 CSS 和 DIV 布局。



图 2-36 任务 2 效果图

① 在“解决方案资源管理器”窗口中右击 SmartOnLine 文件夹,选择“新建文件夹”命令,更改文件夹名为 Shop。

② 右击 Shop 文件夹,选择“添加”→“添加新项”命令,打开“添加新项”对话框,在“名称”对话框中输入 Layout. master,单击“母版页”列表,再单击“添加”按钮。

③ 拖动 HTML 标签页中的 DIV 控件到 Layout. master 母版页中,按照代码中的粗体更改相应的 ID 号和层次。

```
<div id="main_container">
    <div id="header">
    </div>
    <div id="center" class="center_content">
    </div>
    <div id="ft" class="footer">
    </div>
</div>
```

④ 右击 Shop 文件夹,选择“添加”→“添加新项”命令,打开“添加新项”对话框,“名称”文本框中输入 mystyle. css,单击“样式表”列表项,再单击“添加”按钮。打开 mystyle. css 文

件，输入如下代码。

```
body
{
    background:url(images/bg.jpg)repeat-x #e4e9ec top;
    padding:0;
    font-family:Arial, Helvetica, sans-serif;
    font-size:11px;
    margin:0px auto auto auto;
    color:#000;
}
#main_container{
    width:1000px;
    height:auto;
    margin:auto;
    padding:0px;
    background-color:#FFFFFF;
}
#header{
    width:1000px;
    height:136px;
    background:url(images/header_bg.jpg) no-repeat center;
    background-position:0px 0px;
    margin:auto;
}
.center_content
{
    margin:0 auto;
    width:100%;
    float:left;
    /* padding:5px 10px 5px 15px; */
    height:500px;
    text-align:center;
}
.footer{
    width:1000px;
    clear:both;
    height:65px;
    background:url(images/footer_bg.gif) repeat-x top;
}
```

然后在 Layout. master 文件中输入<link rel="stylesheet" type="text/css" href="mystyle. css"/>。这一步主要设置 body 的背景色并使网页居中，同时也设置各个 DIV 容器的样式如长、宽等。

⑤ 打开 mystyle. css 文件，输入如下代码。

```
.top_right{
    width:728px;
    float:right;
}
```

```
.languages{
    float:right;
    width:150px;
    padding:8px 0 0 0;
}
.big_banner{
    float:right;
    padding:10px 10px 0 0;
}
.lang_text{
    float:left;
    padding:0 5px 0 0;
    color:#1DA1CF;
}
#logo{
    float:left;
    padding:45px 0 0 15px;
}
```

这个步骤主要编写相关 CSS 样式以供使用。接着按照下面代码所示的结构层次拖动 DIV 控件,并设置相应的样式(粗体文字表示)。

```
<div id="header">
    <div class="top_right">
        <div class="languages">
            <div class="lang_text">语言:</div>
                <a href="#" class="lang"><img src="images/en.gif" alt=""
                    title="" border="0" /></a>
                <a href="#" class="lang"><img src="images/de.gif" alt=""
                    title="" border="0" /></a>
            </div>
            <div class="big_banner">
                <a href="#"><img src="images/banner728.png" alt=""
                    title="" border="0" /></a>
                </div>
            </div>
            <div id="logo">
                <a href="index.html"><img src="images/logo.png" alt=""
                    title="" border="0" width="182" height="85" /></a>
            </div>
</div>
```

⑥ 拖动 ContentPlaceHolder 控件至 center DIV 部分。此时的效果如图 2-37 所示。

⑦ 接下来进行页脚层的制作。将页脚 ft DIV 分别进行左、中、右布局。拖动 3 个 DIV 控件到 ft 层中,在各个层中编写超链接(代码如下)。

```
<div class="footer">
    <div class="left_footer">
        <img src="images/footer_logo.png" alt="" title="" width="89"
        height="42"/>
    </div>
```

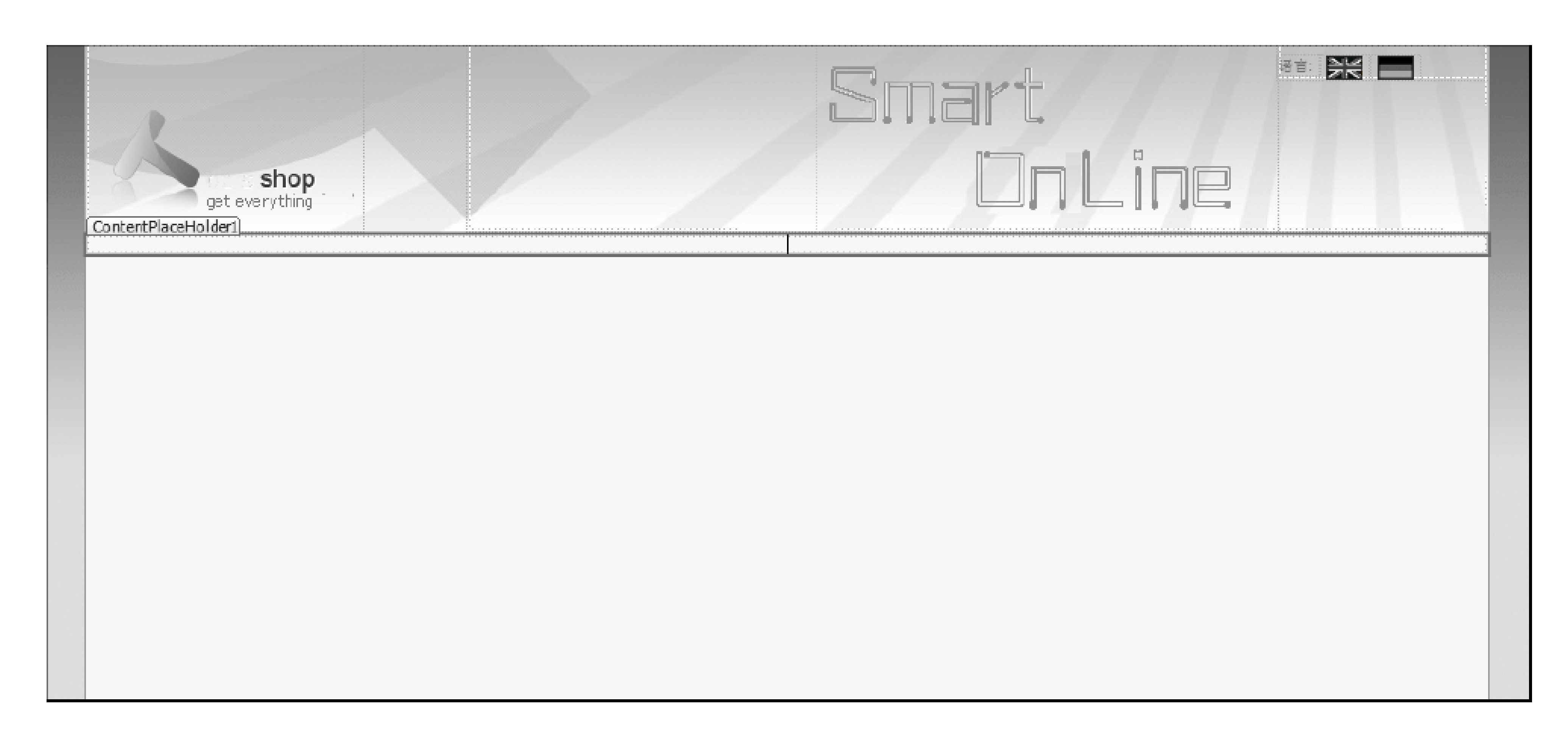


图 2-37　任务 2 效果图 1

```
<div class="center_footer">
        NBCC.CN. All Rights Reserved 2014<br />
        <a href="http://www.nbcc.cn/" title="free css templates">
        www.nbcc.cn</a><br />
        <img src="images/payment.gif" alt="" title="" />
    </div>
    <div class="right_footer">
        <a href="index.html">首页</a>
        <a href="details.html">关于</a>
        <a href="details.html">站点导航</a>
        <a href="details.html">rss</a>
        <a href="contact.html">联系我们</a>
    </div>
</div>
```

⑧ 编写 left_footer、center_footer 和 right_footer 的样式定义。在 mystyle.css 输入以下代码。

```
.footer{
    width:1000px;
    clear:both;
    height:65px;
    background:url(images/footer_bg.gif)repeat-x top;
}
.left_footer{
    float:left;
    width:300px;
    padding:5px 0 0 10px;
}
.right_footer{
    float:right;
    padding:15px 30px 0 0;
}
.right_footer a{
    padding:0 0 0 7px;
```

```
        text-decoration:none;
        color: #666666;
}
.right_footer a:hover{
        text-decoration:underline;
}
```

2.4.3　任务 3：实现 Smart On Line 商城首页的制作

1. 任务目标

(1) 能使用 DIV 和 CSS 实施页面布局。

(2) 能根据母版页创建内容页。

2. 任务内容

(1) 运用 DIV 和 CSS 设计 Smart On Line 商城模板页的布局。

(2) 使用母版页创建新内容页。

3. 任务实施步骤

制作 Smart On Line 商城首页(见图 2-38)分为两个环节。第一个环节是根据母版页创建内容页;第二个环节则是在内容页中使用 CSS 和 DIV 布局(见图 2-39)。

① 右击 Shop 文件夹,选择“添加”→“添加新项”命令,打开“添加新项”对话框,单击“Web 窗体”列表项,选择“选择母版页”复选框,在“名称”文本框中输入 index. aspx。接着单击“添加”按钮,打开“选择母版页”对话框。

② 在“选择母版页”对话框中,选择上一任务完成的 Layout. master 母版页,单击“确定”按钮。

③ 制作导航。拖动 DIV 控件到内容页,设置 id 为 menu_tab,使用 ul 和 li 标签定义无序列表。

```
<div id="menu_tab">
        <ul class="menu">
                <li><a href="index.html" class="nav">首页</a></li>
                <li class="divider"></li>
                <li><a href="WareGrid2.aspx" class="nav">商品</a></li>
                <li class="divider"></li>
                <li><a href="#" class="nav">特价</a></li>
                <li class="divider"></li>
                <li><a href="#" class="nav">我的账户</a></li>
                <li class="divider"></li>
                <li><a href="Login.aspx" class="nav">登录</a></li>
                <li><a href="reg.aspx"class="nav">注册</a></li>
                <li class="divider"></li>
                <li><a href="#" class="nav">快运 </a></li>
                <li class="divider"></li>
                <li><a href="contact.html" class="nav">联系我们</a></li>
        </ul>
</div>
```

然后在 mystyle. css 文件中编写菜单所需的样式,代码如下。



图 2-38　首页效果图



图 2-39　使用 CSS 和 DIV 布局

```
#menu_tab{
     width:1000px;
     height:36px;
     background:url(images/menu_bg.gif) repeat-x;
}
ul.menu {
     list-style-type:none; float:left; display:block; width:982px;
     margin:0px; padding:0px;background:url(images/menu_bg.gif)repeat-x;
}
ul.menu li{
     display:inline;
     font-size:11px;
     font-weight:bold;
     line-height:36px;
}
ul.menu li.divider {
     display:inline;
     width:4px;
     height:36px;
     float:left;
     background:url(images/menu_divider.gif) no-repeat center;
}

a.nav:link, a.nav:visited {
     display:block;
     float:left;
     padding:0px 8px 0px 8px;
       margin:0 14px 0 14px;
     height:36px;
     text-decoration:none;
     color:#fff;
}
a.nav:hover{
     display:block;
     float:left;
     padding:0px 8px 0px 8px;
     margin:0 14px 0 14px;
     height:36px;
     text-decoration:none;
     color:#199ECD;
}
```

④ 按照图 2-39 所示效果图制作左侧网页。鉴于大量的 html 网页制作不是本教程的重点，这里将相应的 HTML 代码和 CSS 代码作下描述，具体类似操作就简略。左侧效果网页代码如下。

```
<div class="crumb_navigation">
          导航：<span class="current">主页</span>
</div>
<div class="left_content">
     <div class="title_box">分类</div>
```

```
        <ul class="left_menu">
        <li class="odd"><a href="details.html">家用电器</a></li>
        <li class="even"><a href="details.html">手机数码</a></li>
        <li class="odd"><a href="details.html">服饰内衣</a></li>
        <li class="even"><a href="details.html">汽车用品</a></li>
        <li class="odd"><a href="details.html">营养保健</a></li>
        <li class="even"><a href="details.html">图书音像</a></li>
        <li class="odd"><a href="details.html">家居家具</a></li>
        <li class="even"><a href="details.html">个性化妆</a></li>
        <li class="odd"><a href="details.html">运动健康</a></li>
        <li class="even"><a href="details.html">食品饮料</a></li>
        <li class="odd"><a href="details.html">彩票旅行</a></li>
        <li class="even"><a href="details.html">珠宝首饰</a></li>
        </ul>
    <div class="title_box">特价</div>
    <div class="border_box">
        <div class="product_title"><a href="details.html">
            欧宝丽(oboni) 32JIM 32英寸 超窄边 LED液晶电视 (黑色)</a></div>
        <div class="product_img"><a href="details.html"><img src=
        "images/p1.jpg"
                alt="" title="" border="0" /></a></div>
        <div class="prod_price"><span class="reduce">350¥</span>
        <span class="price">
                270¥</span></div>
    </div>
    <div class="title_box">推送</div>
    <div class="border_box">
        <input type="text" name="newsletter" class="newsletter_input"
                value="你的 E-mail 地址"/>
        <a href="#" class="join">订阅</a>
    </div>
    <div class="banner_adds">
            <a href="#"><img src="images/bann2.jpg" alt="" title=""
            border="0" /></a>
    </div>
</div>
```

相应的 CSS 代码如下。

```
/* ----------------crumb_navigation------------- */
.crumb_navigation{
    width:980px;
    height:15px;
    padding:5px 10px 0 20px;
    color:#333333;
    background:url(images/navbullet.png) no-repeat left;
    background-position:5px 8px;
}
.crumb_navigation a{
    color:#0fa0dd;
    text-decoration:underline;
}
span.current{
```

```
    color:#0fa0dd;
}

/* -------------left_content------------------ */
.left_content{
    width:180px;
    float:left;
    padding:0 0 0 5px;
}
.title_box{
    width:180px;
    height:30px;
    margin:5px 0 0 0;
    background:url(images/menu_title_bg.gif) no-repeat center;
    text-align:center;
    font-size:13px;
    font-weight:bold;
    color:#159dcc;
    line-height:30px;
}
/* --------left menu--------- */
ul.left_menu{
    width:180px;
    padding:0px;
    margin:0px;
    list-style:none;
}
ul.left_menu li{
    margin:0px;
    list-style:none;
}
ul.left_menu li.odd a{
    width:166px;height:25px;display:block; border-bottom:1px #
    e4e4e4 dashed;
    text-decoration:none;color:#504b4b;padding:0 0 0 14px;
    line-height:25px;
}
ul.left_menu li.even a{
    width:166px;height:25px;
    display:block; border-bottom:1px #e4e4e4 dashed; background-color:
    #f0f4f5;
    text-decoration:none;color:#504b4b;padding:0 0 0 14px;
    line-height:25px;
}
ul.left_menu li.even a:hover, ul.left_menu li.odd a:hover{
    color:#000; text-decoration:underline;
}
.border_box{
    width:180px;
    height:auto;
    text-align:center;
```

```
        background:url(images/box_bottom_bg.gif) no-repeat center bottom;
    }
    .product_title{
        color:#ff8a00;
        padding:5px 0 5px 0;
        font-weight:bold;
    }
    .product_title a{
        text-decoration:none;
        color:#ff8a00;
        padding:5px 0 5px 0;
        font-weight:bold;
    }
    .product_title a:hover{
        color:#064E5A;
    }
    .product_img{
        padding:5px 0 5px 0;
    }
    .prod_price{
        padding:5px 0 5px 0;
    }
    span.reduce{
        color:#666666;
        text-decoration:line-through;
    }
    span.price{
        color: #ff8a00;
    }
    /* ----------newsletter-------------- */
    input.newsletter_input{
        width:150px;
        height:16px;
        border:1px #ddd9d9 solid;
        margin:10px 0 5px 0;
        font-size:12px;
        padding:3px;
        color:#999999;
    }
    a.join{
        width:28px;
        display:block;
        margin:0px 0 5px 110px;
        padding:2px 8px 6px 8px;
        text-decoration: underline;
        color:#169ECC;
    }
    .banner_adds{
        width:180px;
        text-align:center;
        padding:10px 0 10px 0;
```

}

⑤ 制作中间及右侧的效果。限于篇幅，具体代码本书就不一一列举了。请查看本书配套的源代码。网站整体运行效果如图 2-38 所示。

2.4.4　任务 4：实现 Smart On Line 商城首页广告的轮动显示

1. 任务目标

能使用 JavaScript 实现网页的特殊效果。

2. 任务内容

能熟练使用 JavaScript 实现广告轮动显示的效果。

3. 任务实施步骤

实现广告轮显采用 JS 代码实现。搜狐网站中为开发人员提供了封装的 JS 代码供开发人员使用。本书使用其封装的 SohuFlash 实现广告轮显效果(见图 2-40)。



图 2-40　首页广告轮显效果图

① 将该任务所需的素材 17173msw. swf 和 eye. js 文件复制到项目相应目录中。右击 Shop 文件夹，选择“添加”→“新建文件夹”命令，将文件更名为 swf。以同样方式创建 js 文件夹。将 17173msw. swf 复制到 swf 文件夹中，将 eye. js 复制到 js 文件夹中。

② 在上一任务完成的 index. aspx 首页文件的<div class="oferta">后面插入 DIV 代码，设置 id 为 sasFlashFocus311，并插入 Flash 元素，其代码如下所示。该段代码显示常见的 Flash 动画效果的设置参数，其中该任务相关的元素主要是 Flash 动画显示的高和宽以及 Flash 文件名，已经在代码中用粗体进行标识。

```
<DIV id= sasFlashFocus311 >
    <OBJECT id= sasFlashFocus311 height= 156 width= 585
        classid= clsid : D27CDB6E - AE6D - 11cf - 96B8 - 444553540000 >
        <PARAM NAME= "_cx" VALUE= "7990" >
        <PARAM NAME= "_cy" VALUE= "6006" >
        <PARAM NAME= "FlashVars" VALUE= "" >
```

```
        <PARAM NAME="Movie" VALUE="swf/17173msw.swf">
        <PARAM NAME="Src" VALUE="swf/17173msw.swf">
        <PARAM NAME="WMode" VALUE="Opaque">
        <PARAM NAME="Play" VALUE="0">
        <PARAM NAME="Loop" VALUE="-1">
        <PARAM NAME="Quality" VALUE="High">
        <PARAM NAME="SAlign" VALUE="LT">
        <PARAM NAME="Menu" VALUE="0">
        <PARAM NAME="Base" VALUE="">
        <PARAM NAME="AllowScriptAccess" VALUE="">
        <PARAM NAME="Scale" VALUE="NoScale">
        <PARAM NAME="DeviceFont" VALUE="0">
        <PARAM NAME="EmbedMovie" VALUE="0">
        <PARAM NAME="BGColor" VALUE="">
        <PARAM NAME="SWRemote" VALUE="">
        <PARAM NAME="MovieData" VALUE="">
        <PARAM NAME="SeamlessTabbing" VALUE="1">
        <PARAM NAME="Profile" VALUE="0">
        <PARAM NAME="ProfileAddress" VALUE="">
        <PARAM NAME="ProfilePort" VALUE="0">
        <PARAM NAME="AllowNetworking" VALUE="all">
        <PARAM NAME="AllowFullScreen" VALUE="false">
    </OBJECT>
</DIV>
```

③ index.aspx 文件的<head>元素中输入以下代码：

```
<SCRIPT src="js/eye.js" type=text/javascript></SCRIPT>
```

④ 在步骤②中输入的</OBJECT>代码后面输入如下代码，其中 pics 变量代表要轮显的图片，mylinks 对应每个图片被单击时可以访问的链接地址。

```
<SCRIPT type=text/javascript>
    var pic_width=585; //图片宽度
    var pic_height=156; //图片高度
    var button_pos=2; //按钮位置 1左 2右 3上 4下
    var stop_time=2000; //图片停留时间(1000为1秒钟)
    var show_text=0; //是否显示文字标签 1显示 0不显示
    var txtcolor="ff0000"; //文字色
    var bgcolor="DDDDDD"; //背景色
    var swf_height=show_text==1 ? pic_height+20 : pic_height;
    var pics="", mylinks="", texts="";
    pics='images/01.jpg|images/02.jpg|images/03.jpg|
        images/04.jpg|images/05.jpg|images/06.jpg';
    mylinks='http://www.sina.com/|http://www.sina.com/|http://www.sina.com/
        |http://www.sina.com/|http://www.sina.com/|http://www.sina.com/';
    var sohuFlash2=new sohuFlash("swf/17173msw.swf", "sasFlashFocus311",
            pic_width, swf_height, "6");
    sohuFlash2.addParam("quality", "high");
```

```
    sohuFlash2.addParam("wmode", "opaque");
    sohuFlash2.addVariable("pics", pics);
    sohuFlash2.addVariable("links", mylinks);
    sohuFlash2.addVariable("texts", texts);
    sohuFlash2.addVariable("pic_width", pic_width);
    sohuFlash2.addVariable("pic_height", pic_height);
    sohuFlash2.addVariable("show_text", show_text);
    sohuFlash2.addVariable("txtcolor", txtcolor);
    sohuFlash2.addVariable("bgcolor", bgcolor);
    sohuFlash2.addVariable("button_pos", button_pos);
    sohuFlash2.addVariable("stop_time", stop_time);
    sohuFlash2.write("sasFlashFocus311");
</ SCRIPT >
```

25　总结归纳

在 ASP.NET Web 网站开发过程中，必须要运用 HTML 等基本知识。限于篇幅，本书中将不再具体阐述 HTML 的基本知识，想了解 HTML 知识的读者可参考相关书籍。

26　课后习题

选择题

1. 当表单各项填写完毕，单击“提交”按钮时可以触发(　　)事件。

A. onEnter　　B. onSubmit　　C. onMouseDrag　　D. onMouseOver

2. 分析下面的 JavaScript 代码段，输出结果是(　　)。

```
var a=15.59;
document.write(Math.round(a));
```

A. 15　　B. 16　　C. 15.5　　D. 15.4

3. 使用 JavaScript 实现下面的功能：在一个文本框中内容发生改变后，单击页面的其他部分将弹出一个消息框来显示文本框中的内容。下面语句正确的是(　　)。

A. <INPUT TYPE="text" onChange = "alert(this.value) ">

B. <INPUT TYPE="text" onClick = "alert(this.value) ">

C. <INPUT TYPE="text" onChange = "alert(text.value) ">

D. <INPUT TYPE="text" onClick = "alert(value) ">

4. 假定今天是 2006 年 4 月 1 日(星期六)，请问下列 JavaScript 代码在页面上的输出结果是(　　)。

```
var time=new Date();
document.write(time.getDate());
```

A. 2006　　B. 4　　C. 1　　D. 6

5. 在 HTML 中，表单中的 input 元素的(　　)属性用于指定表单元素的名称。

A. value　　B. name　　C. type　　D. caption

6. 下面描述正确的是(　　)。

A. switch 语句用于重复执行一个语句块的操作

B. switch 语句根据表达式的值执行若干语句块之一，如果没有匹配项，则执行默认语句块中的语句

C. switch 语句表达式中的值不能与后面 case 语句中的常量相匹配时将出现运行错误

D. switch 语句又叫循环语句

7. 分析如下 JavaScript 代码，b 的值为(　　)。

```
var a=1.5,b;
b=parseInt(a);
```

A. 2　　B. 0.5　　C. 1　　D. 1.5

8. CSS 指的是(　　)。

A. Computer Style Sheets　　B. Cascading Style Sheets

C. Creative Style Sheets　　D. Colorful Style Sheets

9. 在以下的 HTML 中，正确引用外部样式表的方法是(　　)。

A. <style src="mystyle.css">

B. <link rel="stylesheet" type="text/css" href="mystyle.css">

C. <stylesheet>mystyle.css</stylesheet>

10. 在 HTML 文档中，引用外部样式表的正确位置是(　　)。

A. 文档的末尾　　B. 文档的顶部　　C. <body>部分　　D. <head>部分

11. 用于定义内部样式表的 HTML 标签是(　　)。

A. <style>　　B. <script>　　C. <css>

12. 可用来定义内联样式的 HTML 属性是(　　)。

A. font　　B. class

C. styles　　D. style

13. 下列 CSS 语法中正确的是(　　)。

A. body:color=black　　B. {body:color=blackbody}

C. body {color: black}　　D. {body;color:black}

14. 在 CSS 文件中插入注释的方法是(　　)。

A. // this is a comment　　B. // this is a comment //

C. /* this is a comment */　　D. this is a comment

15. 可用于改变背景颜色的属性是(　　)。

A. bgcolor:A、B、C、

B. background-color:

C. color:

16. 为所有的<h1>元素添加背景颜色的代码是(　　)。
    A. h1. all {background-color:#FFFFFF}
    B. h1 {background-color:#FFFFFF}
    C. all. h1 {background-color:#FFFFFF}
17. 可以改变某个元素的文本颜色的属性是(　　)。
    A. text-color:　　B. fgcolor:
    C. color:　　D. text-color
18. 可控制文本大小的 CSS 属性是(　　)。
    A. font-size　　B. text-style
    C. font-style　　D. text-size
19. 在以下的 CSS 中,可使所有 <p> 元素变为粗体的正确语法是(　　)。
    A. <p style="font-size:bold">　　B. <p style="text-size:bold">
    C. p {font-weight:bold}　　D. p {text-size:bold}
20. 显示没有下划线超链接的方法是(　　)。
    A. a {text-decoration:none}　　B. a {text-decoration:no underline}
    C. a {underline:none}　　D. a {decoration:no underline}

27 同步操练

在项目 1 中已经完成了格林酒店管理系统的数据库设计,项目经理指定设计并创建格林酒店网站首页,现要求使用 Visual Studio. NET 2012 开发工具创建 GreenHotel 网站,并更改标题为"欢迎光临格林连锁夏日酒店",最终效果如图 2-41 所示。



图 2-41　格林酒店首页效果图

项目 3　会 员 注 册

3.1　项 目 引 入

会员注册是 Smart On Line 电子商城的必不可少的一项功能，也是进行客户分析数据的来源之一。李明是 Smart On Line 电子商城的一位开发人员，现接到任务，要求协助开发用户注册功能模块。

3.2　项 目 分 析

经过小组讨论和仔细分析，认为 Smart On Line 电子商城用户注册功能可以使用 ASP. NET常见的服务器控件来设计界面；此外还认为，为使客户拥有良好的体验，对界面输入数据的合法性应进行验证；在验证功能中还须提供验证码功能以防恶意注册。最后当用户单击“注册”按钮时，将用户输入信息保存到数据库中。因此，项目经理将这个项目分为三个任务，一是用户注册界面设计，二是用户输入信息的验证，最后是保存注册用户信息。

3.3　知 识 准 备

3.3.1　ASP. NET 常用服务器控件

每个控件都有一些公共属性，例如字体颜色、边框的颜色、样式等。在 Visual Studio 2012 中，当开发人员用鼠标选择了相应的控件后，属性栏中会简单地显示该属性的作用，如图 3-1 所示。

属性栏用来设置控件的属性，当控件在页面中被初始化时，这些将被应用到控件中。控件的属性也可以通过编程的方法在页面相应代码区域编写，示例代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Visible= false;
}
```

上述代码编写了一个 Page_Load(页面加载)事件，当页面初次被加载时，会执行 Page_Load 事件中的代码。这里通过编程的方法对控件的属性进行了更改。当页面加载时，控件的属性会被应用并呈现在浏览器中。

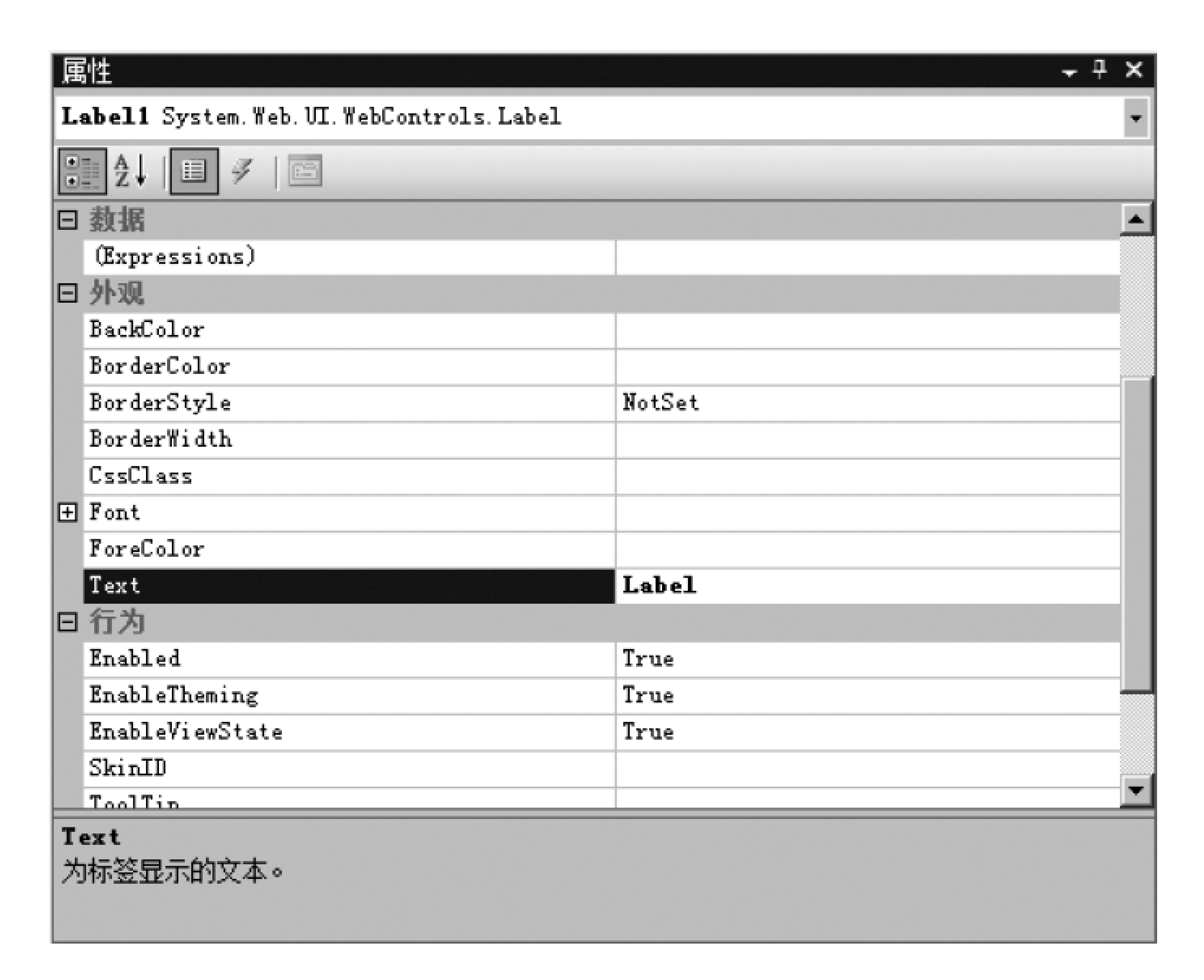


图 3-1　控件的属性

ASP. NET 提供了诸多控件，这些控件包括简单控件、数据库控件、登录控件等强大的控件。在 ASP. NET 中，简单控件是最基础也是经常被使用的控件，简单控件包括标签控件(Label)、超链接控件(HyperLink)以及图像控件(Image)等。

1. 标签控件(Label)

在 Web 应用中，希望显示的文本不能被用户更改；或者当触发事件时，某一段文本能够在运行时更改，则可以使用标签控件(Label)。开发人员可以非常方便地将标签控件拖放到页面中，该页面将自动生成一段标签控件的声明代码，示例代码如下。

```
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
```

上述代码中声明了一个标签控件，并将这个标签控件的 ID 属性设置为默认值 Label1。由于该控件是服务器端控件，所以在控件属性中包含“runat="server"”属性。该代码还将标签控件的文本初始化为 Label，开发人员能够配置该属性并进行不同文本内容的呈现。同样，标签控件的属性能够在相应的. cs 代码中初始化，示例代码如下所示。

```
protected void Page_PreInit(object sender,EventArgs e)
{
    Label1.Text="Hello World";
}
```

上述代码在页面初始化时将 Label1 的文本属性设置为“Hello World”。

2. 超链接控件

超链接控件(HyperLink)相当于实现了 HTML 代码中的“”显示的效果，当然，超链接控件有自己的特点，当拖动一个超链接控件到页面中时，系统会自动生成控件声明代码，示例代码如下所示。

```
<asp:HyperLink ID="HyperLink1" runat="server">HyperLink</asp:HyperLink>
```

上述代码声明了一个超链接控件，相对于 HTML 代码形式，超链接控件可以通过传递

指定的参数来访问不同的页面。当触发了一个事件后，超链接的属性可以被改变。超链接控件通常使用的两个属性如下。

- ImageUrl：要显示的图像的 URL。
- NavigateUrl：要跳转的 URL。

（1）ImageUrl 属性

设置 ImageUrl 属性可以设置这个超链接是以文本形式显示还是以图片文件显示，虽然表现形式不同，但是不管是图片形式还是文本形式，全都实现相同的效果。

（2）Navigate 属性

Navigate 属性可以为无论是文本形式还是图片形式的超链接设置超链接属性，即将要跳转的页面，示例代码如下所示。

```
<asp:HyperLink ID="HyperLink1" runat="server"
    ImageUrl="~/Shop/Images/log.gif"
    NavigateUrl="~/Shop/About.aspx">
    HyperLink
</asp:HyperLink>
```

上述代码使用了图片超链接的形式。当单击此超链接控件后，浏览器将跳转到名为 About.aspx 的页面中。

（3）动态跳转

在前面讲解了超链接控件的优点，超链接控件的优点在于能够对控件进行编程，来按照用户的意愿跳转到自己跳转的页面。以下代码实现了当用户选择 Sina 时，会跳转到 Sina 网站；如果选择 Sohu，则会跳转到 Sohu 页面。示例代码如下所示。

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (DropDownList1.Text == "sina")
    {
        HyperLink1.NavigateUrl= "http://www.sina.com";
    }
    else
    {
        HyperLink1.NavigateUrl= "http://www.sohu.com";
    }
}
```

上述代码使用了 DropDownList 控件，当用户选择不同的值时，对 HyperLink1 控件进行操作。值得注意的是，如果只是为了单纯地实现超链接，同样不推荐使用 HyperLink 控件，因为过多的使用服务器控件同样有可能造成性能问题。

3. 图像控件

图像控件(Image)用来在 Web 窗体中显示图像，图像控件常用的属性如下。

- AlternateText：在图像无法显示时，显示备用的文本。
- ImageAlign：图像的对齐方式。
- ImageUrl：要显示图像对应的 URL。

当图片无法显示的时候，图片将被替换成 AlternateText 属性中的文字，ImageAlign 属

性用来控制图片的对齐方式，而 ImageUrl 属性用来设置图像连接地址。同样，HTML 中也可以使用＜img src＝" " "alt＝" "＞来替代图像控件。图像控件具有可控性的优点，就是通过编程可以控制图像控件。除了显示图形以外，Image 控件的其他属性还允许为图像指定各种文本，如 GenerateEmptyAlternateText 属性，如果将此属性设置为 True，则要呈现的图片的 alt 属性将设置为空。

4. 文本框控件

在 Web 开发中，Web 应用程序通常需要和用户进行交互，例如用户注册、登录、发帖等，那么就需要文本框控件（TextBox）来接受用户输入的信息。开发人员还可以使用文本框控件制作高级的文本编辑器并用于 HTML，以及文本的输入/输出。通常情况下，默认的文本框控件（TextBox）是一个单行的文本框，用户只能在文本框中输入一行内容。通过修改该属性，则可以将文本框设置为多行或者以密码的形式显示，文本框控件常用的控件属性如下。

- AutoPostBack：在文本修改以后，是否自动回传。
- Columns：文本框的宽度。
- EnableViewState：控件是否自动保存其状态以用于往返过程。
- MaxLength：用户输入的最大字符数。
- ReadOnly：是否为只读。
- Rows：作为多行文本框时所显示的行数。
- TextMode：文本框的模式，设置单行、多行、密码和数字等。
- Wrap：文本框是否换行。

（1）AutoPostBack（自动回传）属性

在网页的交互中，如果用户提交了表单，或者执行了相应的方法，那么该页面将会发送到服务器上，服务器将执行表单的操作或者执行相应方法后，再呈现给用户，例如按钮控件、下拉菜单控件等。如果将某个控件的 AutoPostBack 属性设置为 True 时，则如果该控件的属性被修改，那么同样会使页面自动发回到服务器。

（2）EnableViewState（控件状态）属性

ViewState 是 ASP.NET 中用来保存 Web 控件回传状态的一种机制，它是由 ASP.NET 页面框架管理的一个隐藏字段。在回传发生时，ViewState 数据同样将回传到服务器，ASP.NET 框架解析 ViewState 字符串并为页面中的各个控件填充该属性。而填充后，控件通过使用 ViewState 将数据重新恢复到以前的状态。在使用某些特殊的控件时，如数据库控件，来显示数据库。每次打开页面执行一次数据库往返过程是非常不明智的。开发人员可以绑定数据，在加载页面时仅对页面设置一次，在后续的回传中，控件将自动从 ViewState 中重新填充，减少了数据库的往返次数，从而不使用过多的服务器资源。在默认情况下，EnableViewState 的属性值通常为 True。

（3）其他属性

上面的两个属性是比较重要的属性，其他的属性也经常使用。

- MaxLength：在注册时可以限制用户输入的字符串长度。
- ReadOnly：如果将此属性设置为 True，那么文本框内的值是无法被修改的。
- TextMode：此属性可以设置文本框的模式，例如单行、多行和密码形式。默认情况下，不设置 TextMode 属性，那么文本框默认为单行。

文本框无论是在 Web 应用程序开发还是 Windows 应用程序开发中都是非常重要的。文本框在用户交互中能够起到非常重要的作用。在文本框的使用中，通常需要获取用户在文本框中输入的值或者检查文本框属性是否被改写。当获取用户的值的时候，必须通过一段代码来控制。例如当用户单击按钮控件时，需要实现标签控件的文本改变。为了实现相应的效果，可以通过编写 cs 文件代码进行逻辑处理，示例代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)                    //双击按钮时触发的事件
{
    Label1.Text=TextBox1.Text;
}
```

上述代码中，当双击按钮时，就会触发一个按钮事件，这个事件就是将文本框内的值赋值到标签内。同样，双击文本框控件，会触发 TextChange 事件。而当运行时，当文本框控件中的字符变化后，并没有自动回传，这是因为在默认情况下，文本框的 AutoPostBack 属性被设置为 False。当 AutoPostBack 属性被设置为 True 时，文本框的属性变化，则会发生回传，示例代码如下所示。

```
protected void TextBox1_TextChanged(object sender, EventArgs e)
{
    Label1.Text=TextBox1.Text;
}
```

上述代码中，为 TextBox1 添加了 TextChanged 事件。在 TextChanged 事件中，并不是每一次文本框的内容发生了变化之后，就会重传到服务器，这一点和 WinForm 是不同的，因为这样会大大降低页面的执行效率。而当用户将文本框中的焦点移出导致 TextBox 会失去焦点时，才会发生重传。

5. 按钮控件（Button，LinkButton，ImageButton）

在 Web 应用程序和用户交互时，常常需要提交表单、获取表单信息等操作，因此按钮控件是非常必要的。按钮控件能够触发事件，或者将网页中的信息回传给服务器。在 ASP.NET 中，包含三类按钮控件，分别为 Button、LinkButton、ImageButton。

按钮控件的通用属性说明如下。

按钮控件用于事件的提交，按钮控件包含一些通用属性，按钮控件的常用通用属性包括以下几种。

- Causes Validation：按钮是否导致激发验证检查。
- CommandArgument：与此按钮管理的命令参数。
- CommandName：与此按钮关联的命令。

按钮控件对应的事件通常是 Click（单击）和 Command（命令）事件。在 Click 事件中，通常用于编写用户单击按钮时所需要执行的事件。按钮控件中，Click 事件并不能传递参数，所以处理的事件相对简单。而 Command 事件可以传递参数，负责传递参数的是按钮控件的 CommandArgument 和 CommandName 属性。例如将 CommandArgument 和 CommandName 属性设置为 Show，创建一个 Command 事件并在事件中编写相应代码，示例代码如下所示。

```
protected void Button1_Command(object sender, CommandEventArgs e)
{
```

```
        if (e.CommandName == "Show")
        {
            Label1.Text=e.CommandArgument.ToString();
        }
    }
```

需要注意的是，当按钮同时包含 Click 和 Command 事件时，通常情况下会执行 Command 事件。Command 有一些 Click 不具备的好处，就是传递参数。可以对按钮的 CommandArgument 和 CommandName 属性分别设置，通过判断 CommandArgument 和 CommandName 属性来执行相应的方法。这样一个按钮控件就能够实现不同的方法，使得多个按钮与一个处理代码关联或者一个按钮根据不同的值进行不同的处理和响应。相比 Click 单击事件而言，Command 命令事件具有更高的可控性。

6. 单选控件和单选组控件（RadioButton 和 RadioButtonList）

在投票等系统中，通常需要使用单选控件和单选组控件。顾名思义，在单选控件和单选组控件的项目中，只能在有限种选择中进行一个项目的选择。在进行投票等应用开发并且只能在选项中选择单项时，单选控件和单选组控件都是最佳的选择。单选控件可以为用户选择某一个选项，单选控件常用属性如下所示。

- Checked：控件是否被选中。
- GroupName：单选控件所处的组名。
- TextAlign：文本标签相对于控件的对齐方式。

单选控件通常需要 Checked 属性来判断某个选项是否被选中，多个单选控件之间可能存在着某些联系，这些联系通过 GroupName 进行约束和联系。与 TextBox 文本框控件相同的是，单选控件不会自动进行页面回传，必须将 AutoPostBack 属性设置为 True 时才能在焦点丢失时触发相应的 CheckedChanged 事件。

与单选控件相同，单选组控件也是只能选择一个项目的控件，而与单选控件不同的是，单选组控件没有 GroupName 属性，但是却能够列出多个单选项目。另外，单选组控件所生成的代码也比单选控件实现的相对较少。

单选组控件还包括一些属性，用于样式和重复的配置。单选组控件的常用属性如下。

- DataMember：在数据集用作数据源时做数据绑定。
- DataSource：向列表填入项时所使用的数据源。
- DataTextFiled：提供项文本的数据源中的字段。
- DataTextFormat：应用于文本字段的格式。
- DataValueFiled：数据源中提供项值的字段。
- Items：列表中项的集合。
- RepeatColumn：用于布局项的列数。
- RepeatDirection：项的布局方向。
- RepeatLayout：确定是否在某个表或者流中重复。

与单选控件一样，双击单选组控件时系统会自动生成该事件的声明，同样可以在该事件中确定代码。当选择一项内容时，提示用户所选择的内容，示例代码如下所示。

```
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
```

```
{
    Label1.Text=RadioButtonList1.Text;
}
```

7. 复选框控件和复选组控件(CheckBox 和 CheckBoxList)

当一个投票系统需要用户能够选择多个选择项时,则单选框控件就不符合要求了。ASP. NET 还提供了复选框控件和复选组控件来满足多选的要求。复选框控件和复选组控件与单选框控件和单选组控件一样,都是通过 Checked 属性来判断是否被选择。而不同的是,复选框控件没有 GroupName 属性。对于复选框而言,用户可以在复选框控件中选择多个选项,所以就没有必要为复选框控件进行分组。在单选框控件中,相同组名的控件只能选择一项用于约束多个单选框中的选项,而复选框就没有约束的必要。

与单选组控件相同,为了方便复选控件的使用,. NET 服务器控件中同样包括了复选组控件,拖动一个复选组控件到页面可以同单选组控件一样添加复选组列表。复选组控件最常用的是 SelectedIndexChanged 事件。当控件中某项的选中状态被改变时,则会触发该事件。通常 CheckBoxList1. Items[i]. Selected 是用来判断某项是否被选中,其中 Item 数组是复选组控件中项目的集合,其中 Items[i]是复选组中的第 i 个项目。

注意:复选组控件与单选组控件不同的是,不能够直接获取复选组控件某个选中项目的值,因为复选组控件返回的是第一个选择项的返回值,只能够通过 Item 集合来获取选择某个或多个选中的项目值。

8. 列表控件(DropDownList,ListBox 和 BulletedList)

在 Web 开发中,经常会需要使用列表控件,让用户的输入更加简单。例如在用户注册时,用户的所在地是有限的集合,而且用户不喜欢经常输入,这样就可以使用列表控件。同样,列表控件还能够简化用户输入并且防止用户输入在实际中不存在的数据,如性别的选择等。列表控件能在一个控件中为用户提供多个选项,同时又能够避免用户输入错误的选项。例如,在用户注册时,可以选择性别是“男”或者“女”可以使用 DropDownList 列表控件,同时又避免了用户输入其他的信息。因为性别除了“男”就是“女”,输入其他的信息说明这个信息是错误或者是无效的。下列语句声明了一个 DropDownList 列表控件,示例代码如下所示。

```
<asp:DropDownList ID="DropDownList1" runat="server">
    <asp:ListItem>1</asp:ListItem>
    <asp:ListItem>2</asp:ListItem>
    <asp:ListItem>3</asp:ListItem>
    <asp:ListItem>4</asp:ListItem>
    <asp:ListItem>5</asp:ListItem>
    <asp:ListItem>6</asp:ListItem>
    <asp:ListItem>7</asp:ListItem>
</asp:DropDownList>
```

上述代码创建了一个 DropDownList 列表控件,并手动增加了列表项。同时 DropDownList 列表控件也可以绑定数据源控件。DropDownList 列表控件最常用的事件是 SelectedIndexChanged,当 DropDownList 列表控件选择项发生变化时,则会触发该事件。当用户选择相应的项目时,就会触发 SelectedIndexChanged 事件,开发人员可以通过捕捉相应的用户选中的控件进行编程处理,这里就捕捉了用户选择的数字进行字体大小的更改。

示例 1：宁波城市学院对学生就业满意度进行调查，需要开发调查问卷页面（如图 3-2所示）。通过对该图所示效果图分析，发现该界面主要由文本框、下拉列表、单选列表、多选列表等控件设计而成。

宁波城市学院毕业生跟踪调查问卷
（毕业生填写）

姓　名：　　性　别：女　　学　号：
所学专业：计算机信息管理　　毕业时间：　　联系方式：电话
工作单位：　　所从事的岗位：
请各位同学为了母校的发展，配合进行这项工作，以下题目没有特殊说明的皆为单选题，请认真填写，非常感谢！

1. 您目前的行政职务是什么？
○单位负责人 ○单位中层领导 ○单位科室负责人 ○普通员工
2. 您对目前的工作情况感到满意吗？
○很满意 ○较满意 ○有个工作不容易，凑合着 ○不满意，还想调换
3. 毕业参加工作后，您在工作中的表现如何？
○得到晋升，受到重用 ○收到表彰和好评 ○表现一般 ○不适合目前的工作
4. 您认为自己是否能胜任目前的工作?
○能胜任 ○基本能胜任 ○不能胜任
5. 您认为自己能适应目前的工作的最重要原因是？
○专业对口 ○实际工作能力强 ○人际关系好 ○工作态度端正
6. 您目前的工作和所学专业是否对口？
○对口 ○基本对口 ○不对口
7. 择业时，您考虑过自己专业和岗位是否对口的问题吗？
○考虑过，要与本专业对口 ○考虑过，但不对口也没关系 ○没考虑过
8. 毕业后，您调换过几次单位？
○没有调换 ○一次 ○两次 ○三次 ○三次以上
9. 您调换工作的原因是？
○专业不对口 ○工作强度太大 ○与同事关系不融洽 ○单位裁员或倒闭 ○领导不重视 ○经济待遇不好
10. 您认为在学校所学专业与实际工作中的联系程度紧密吗？
○很紧密 ○较紧密 ○不紧密 ○很不紧密
11. 工作中是否遇到因专业知识不精通而不能胜任工作的问题？
○有 ○很少 ○没有
12. 您认为自己所学的专业在企业单位中的认可度怎样？
○十分认可 ○比较认可 ○不认可
13. 您如何看待自己的工作发展前景？
○十分乐观，有所作为 ○比较看好 ○悲观，没有前景 ○没有考虑过
14. 您觉得现在最需要加强哪方面知识的充实？
○人际关系 ○专业知识 ○外语 ○专业技能 ○其他
15. 在校期间，哪些教学环节对您现在的工作帮助最大？
○专业理论课 ○公共基础课 ○实验操作学习 ○职业规划设计 ○课外学术交流活动 ○其他
16. 工作后，您对母校学习期间所学课程的体会是？
○学得不够深和不够宽广 ○学得不够活，能力培养不足 ○学得很多知识，对现在工作帮助不大 ○学的知识对现在工作帮助很大
17. 您认为自己在校期间取得的技能证书在工作中起到的作用怎样？
○帮助很大 ○基本没用，不代表个人实际能力 ○不被认可
18. 与其他高校毕业的同学比较起来，您认为你从母校学习到的哪方面能力较强？
○思想政治素质 ○专业基础知识 ○实际工作能力 ○学习能力 ○创新能力 ○人际关系处理能力
19. 您下一步的打算是？
○进一步学习，提高自己学历，为长远打算 ○继续做下去，在实践中提高自己 ○走一步是一步，没明确目标 ○跳槽寻找更好的
20. 您如何评价母校对您的培养工作？
○很满意 ○基本满意 ○不太满意 ○不满意
21. 结合实际工作经历，您对学校有何建议（教学、管理、就业指导等）？

提　交

图 3-2　调查问卷效果图

① 拖动 HTML 标签中的 Table 控件到页面，右击 Table 控件，选择“插入”→“右侧列”命令来添加列。

② 按照图 3-2 所示布局，拖动 TextBox 控件、DropDownList 控件到页面合适位置。

③ 单击“所学专业”下拉列表智能提示标识，单击“编辑项”菜单项，打开“集合编辑器”对话框(如图 3-3 所示)。

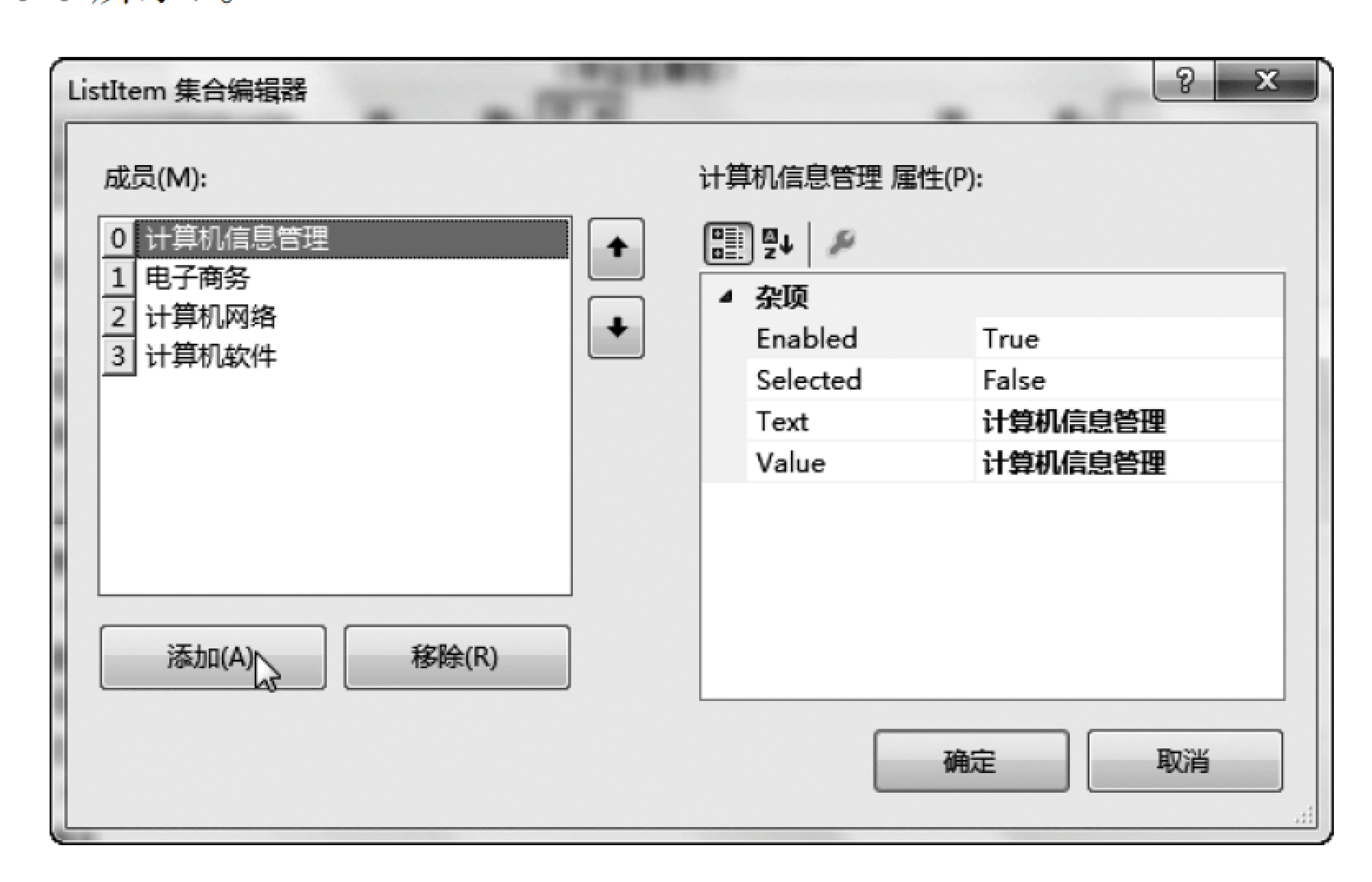


图 3-3　“集合编辑器”对话框

④ 单击“添加”按钮，添加项，修改 Text 属性和 Value 属性(见图 3-3)。

⑤ “性别下拉”列表框和“联系方式”下拉列表的设置同步骤④。

⑥ 拖动 RadioButtonList 到页面，单击智能提示按钮，单击“编辑项”菜单项，打开“集合编辑器”对话框，如同步骤④一样设置显示文本和显示值。接着设置 RadioButtonList 按钮的 RepeatDirection 属性值为 Horizontal，让其水平显示。其中的 Q9 多选列表设置也同 RadioButtonList 设置方法一样。

⑦ 拖动 TextBox 控件到 Q21 下面，设置 TextMode 属性为 Multiline，允许多行输入。

⑧ 拖动 Button 控件到页面，设置其 Text 属性为“按钮”。

3.3.2　ASP.NET 输入验证控件

ASP.NET 提供了强大的验证控件，它可以验证服务器控件中用户的输入，并在验证失败的情况下显示一条自定义错误消息。验证控件直接在客户端执行，用户提交后执行相应的验证，无需使用服务器端进行验证操作，从而减少了服务器与客户端之间的往返过程。

1. 必填验证控件(RequiredFieldValidator)

在实际的应用中，如在用户填写表单时，有一些项目是必填项，例如用户名和密码。在传统的 ASP 中，当用户填写表单后，页面需要发送到服务器并判断表单中的某项 HTML 控件的值是否为空，如果为空，则返回错误信息。在 ASP.NET 中，系统提供了 RequiredFieldValidator 验证控件进行验证。使用 RequiredFieldValidator 控件能够指定某个用户在特定的控件中必须提供相应的信息。如果不填写相应的信息，RequiredFieldValidator 控件就会提示错误信息。RequiredFieldValidator 控件示例代码如下。

```
<body>
    <form id="form1" runat="server">
    <div>
        姓名:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
            runat="server"
            ControlToValidate="TextBox1"
                    ErrorMessage="必填字段不能为空"></asp:
                    RequiredFieldValidator>
        <br />
        密码:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Button" />
        <br />
    </div>
    </form>
</body>
```

在进行验证时,RequiredFieldValidator 控件必须绑定一个服务器控件,在上述代码中,验证控件 RequiredFieldValidator 的服务器控件绑定为 TextBox1,当 TextBox1 中的值为空时,则会提示自定义错误信息为“必填字段不能为空”。

2. 比较验证控件(CompareValidator)

比较验证控件对照特定的数据类型来验证用户的输入。因为当用户输入信息时,难免会输入错误信息,比如当需要了解用户的生日时,很可能输入了其他的字符串。CompareValidator 比较验证控件能够比较控件中的值是否符合开发人员的需要。CompareValidator 控件的特有属性如下。

- ControlToCompare:以字符串形式输入的表达式。要与另一控件的值进行比较。
- Operator:要使用的比较。
- Type:比较两个值的数据类型。
- ValueToCompare:以字符串形式输入的表达式。

当使用 CompareValidator 控件时,可以方便地判断用户是否正确输入了相关值,示例代码如下。

```
<body>
    <form id="form1" runat="server">
    <div>
        请输入生日:
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        毕业日期:
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
         <asp:CompareValidator ID="CompareValidator1" runat="server"
            ControlToCompare="TextBox2" ControlToValidate="TextBox1"
            CultureInvariantValues="True" ErrorMessage="输入格式错误,请改正!"
            Operator="GreaterThan"
            Type="Date">
            </asp:CompareValidator>
```

```
            <br />
            <asp:Button ID="Button1" runat="server" Text="Button" />
            <br />
        </div>
        </form>
</body>
```

上述代码判断 TextBox1 的输入的格式是否正确，当输入的格式错误时，会提示输入了错误的信息，如图 3-4 所示。



图 3-4 CompareValidator 验证控件

CompareValidator 验证控件不仅能够验证输入的格式是否正确，还可以验证两个控件之间的值是否相等。如果两个控件之间的值不相等，CompareValidator 验证控件同样会将自定义错误信息呈现在用户的客户端浏览器中。

3. 范围验证控件(RangeValidator)

范围验证控件(RangeValidator)可以检查用户的输入是否在指定的上限与下限之间。通常情况下用于检查数字、日期、货币等。范围验证控件(RangeValidator)控件的常用属性如下。

- MinimumValue：指定有效范围的最小值。
- MaximumValue：指定有效范围的最大值。
- Type：指定要比较的值的数据类型。

通常情况下，为了控制用户输入的范围，可以使用该控件。当输入用户的生日时，今年是 2008 年，那么用户就不应该输入 2009 年。同样，基本上没有人的寿命会超过 100，所以对输入的日期的下限也需要进行规定，示例代码如下。

```
<div>
        请输入生日:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RangeValidator ID="RangeValidator1" runat="server"
            ControlToValidate="TextBox1" ErrorMessage="超出规定范围,请重新填写"
            MaximumValue="2009/1/1" MinimumValue="1990/1/1" Type="Date">
            </asp:RangeValidator>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Button" />
</div>
```

上述代码将 MinimumValue 属性值设置为 1990/1/1，并将 MaximumValue 的值设置为 2009/1/1，当用户的日期低于最小值或高于最高值时，则提示错误。RangeValidator 验证控件在进行控件值的范围设定时，其范围不仅仅可以是一个整数值，同样还能够是时间、日期等值。

4. 正则验证控件(RegularExpressionValidator)

在上述控件中虽然能够实现一些验证，但是验证的能力是有限的，例如在验证的过程中，只能验证是否是数字，或者是否是日期。也可能在验证时，只能验证一定范围内的数值，虽然这些控件提供了一些验证功能，但却限制了开发人员进行自定义验证和错误信息的开发。为实现一个验证，很可能需要多个控件同时搭配使用。

正则验证控件就解决了这个问题,正则验证控件的功能非常强大,它用于确定输入的控件的值是否与某个正则表达式所定义的模式相匹配,如电子邮件、电话号码以及序列号等。

正则验证控件常用的属性是 ValidationExpression,它用来指定用于验证的输入控件的正则表达式。客户端的正则表达式验证语法和服务端的正则表达式验证语法不同,因为在客户端使用的是 Jsript 正则表达式语法,而在服务器端使用的是 Regex 类提供的正则表达式语法。使用正则表达式能够实现强大字符串的匹配并验证用户输入的格式是否正确,系统提供了一些常用的正则表达式,开发人员能够选择相应的选项进行规则筛选,如图 3-5 所示。

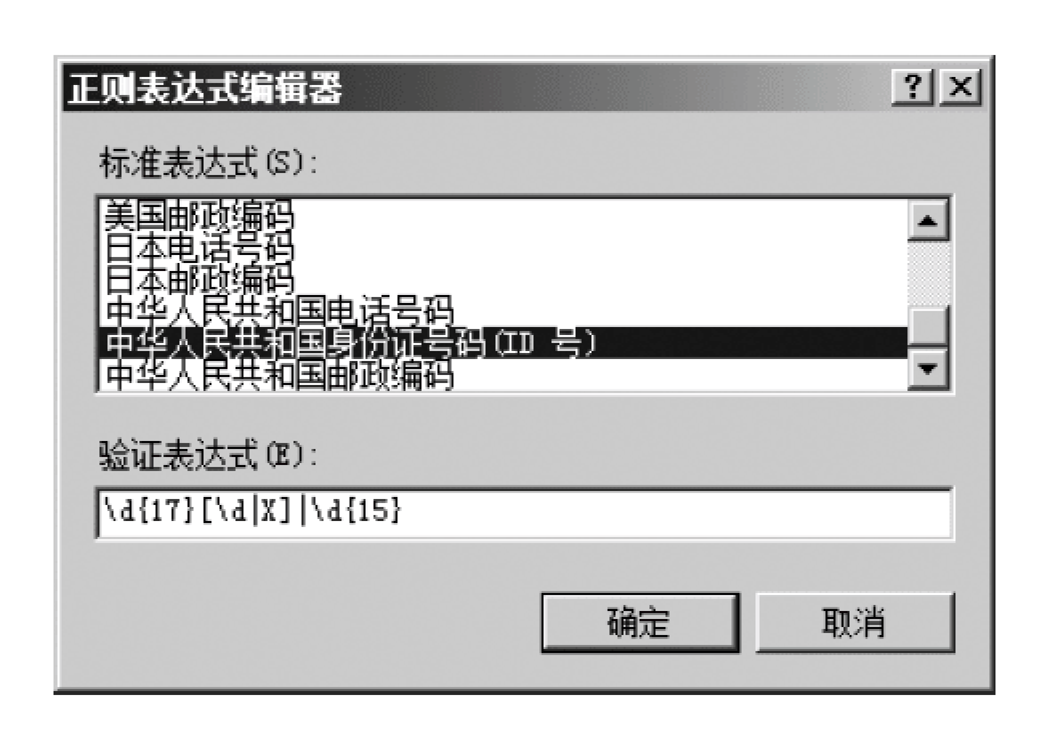


图 3-5 系统提供的正则表达式

当选择了正则表达式后,系统自动生成相应的 HTML 代码。运行后当用户单击按钮控件时,如果输入的信息与相应的正则表达式不匹配,则会提示错误信息。同样,开发人员也可以自定义正则表达式来规范用户的输入。使用正则表达式能够加快验证速度并在字符串中快速匹配,而另一方面,使用正则表达式能够减少复杂的应用程序的功能开发和实现。

注意: 在用户输入为空时,其他的验证控件都会验证通过。所以,在验证控件的使用中,通常需要与表单验证控件(RequiredFieldValidator)一起使用。

5. 自定义逻辑验证控件(CustomValidator)

自定义逻辑验证控件允许使用自定义的验证逻辑创建验证控件。例如.aspx 文件中声明了两个 Label 控件,一个 TextBox 控件,一个 Button 控件,以及一个 CustomValidator 控件。user()函数可检测输入值的长度。如果长度小于 8 或大于 16,将在 CustomValidator 控件中显示文本"用户名必须介于 8 到 16 个字符之间!"。其网页代码如下。

```
<html>
    <body>
        <form runat="server">
            <asp:Label runat="server" Text="请输入用户名:" />
            <asp:TextBox id="txt1" runat="server" />
            <asp:Button Text="提交" runat="server"/>
            <br />
            <asp:Label id="mess" runat="server"/>
            <br />
            <asp:CustomValidator
            ControlToValidate="txt1"
            OnServerValidate="user"
            Text="用户名必须介于 8 到 16 个字符之间!"
            runat="server"/>
        </form>
    </body>
</html>
```

其自定义验证代码如下。

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
```

```
{
    if(len(args.Value)<8 || len(args.Value)>16)
        args.IsValid=false;
    else
        args.IsValid=true;
}
```

从 CustomValidator 验证控件的验证代码可以看出，CustomValidator 验证控件可以在服务器上执行验证检查。如果要创建服务器端的验证函数，则处理 CustomValidator 控件的 ServerValidate 事件。使用传入的 ServerValidateEventArgs 的对象的 IsValid 字段来设置是否通过验证。

而 CustomValidator 控件同样也可以在客户端实现，该验证函数可用 VBScript 或 Jscript 来实现，而在 CustomValidator 控件中需要使用 ClientValidationFunction 属性指定与 CustomValidator 控件相关的客户端验证脚本的函数名称进行控件中的值的验证。

6. 验证组控件（ValidationSummary）

验证组控件能够对同一页面的多个控件进行验证。同时，验证组控件通过 ErrorMessage 属性为页面上的每个验证控件显示错误信息。验证组控件的常用属性如下。

- DisplayMode：可显示为列表、项目符号列表或单个段落。
- HeaderText：指定一个自定义标题。
- ShowMessageBox：确定是否在消息框中显示摘要。
- ShowSummary：确定是显示还是隐藏 ValidationSummary 控件。

验证组控件能够显示页面的多个控件产生的错误。

3.3.3　ADO. NET 数据访问模型

ADO. NET 是. NET Framework 中的一系列类库，它能够让开发人员更加方便地在应用程序中使用和操作数据。在 ADO. NET 中，大量复杂的数据操作的代码被封装起来，所以开发人员在 ASP. NET 应用程序开发中只需要编写少量的代码即可处理大量的操作。ADO. NET 与 C#. NET、VB. NET 不同的是，ADO. NET 并不是一种语言，而是对象的集合。ADO. NET 由微软编写代码，提供了在. NET 开发中数据库所需要的操作类。在. NET应用程序开发中，C# 和 VB. NET 都可以使用 ADO. NET。ADO. NET 可以被看作是一个介于数据源和数据使用者之间的转换器。ADO. NET 接受使用者语言中的命令，如连接数据库、返回数据集之类，然后将这些命令转换成在数据源中可以正确执行的语句。在传统的应用程序开发中，应用程序可以连接 ODBC 来访问数据库，虽然微软提供的类库非常丰富，但是开发过程却并不简单。ADO. NET 在另一方面可以说简化了这个过程。用户无需了解数据库产品的 API 或接口，也可以使用 ADO. NET 对数据进行了操作。ADO. NET中常用的对象如下。

- SqlConnection：该对象表示与数据库服务器进行连接。
- SqlCommand：该对象表示要执行的 SQL 命令。
- SqlParameter；该对象代表了一个将被命令中标记代替的值。
- SqlDataAdapter：该对象表示填充命令中的 DataSet 对象的能力。
- DataSet：表示命令执行的结果，可以是数据集，并且可以与 BulletedList 进行绑定。

通过使用上述对象,可以轻松地连接数据库并对数据库中的数据进行操作。对开发人员而言,可以使用 ADO. NET 对数据库进行操作,在 ASP. NET 中,还提供了高效的控件,这些控件同样使用了 ADO. NET 让开发人员能够连接、绑定数据集并进行相应的数据操作。

1. 连接 SQL 数据库

ADO. NET 通过 ADOConnection 连接到数据库。与 ADO 的 Connection 对象相似的是,ADOConnection 同样包括 Open 和 Close 方法。Open 表示打开数据库连接,Close 表示关闭数据库连接。在每次打开数据库连接后,都需要关闭数据库连接。

(1) 建立连接

在 SQL 数据库的连接中,需要使用. NET 提供的 SqlConnection 对象来对数据库进行连接。在连接数据库前,需要为连接设置连接串,连接串就相当于告诉应用程序怎样找到数据库去进行连接,然后程序才能正确地与 SQL 建立连接,连接字串示例代码如下所示。

```
server="服务器地址";database="数据库名称";uid="数据库用户名";pwd="数据库密码";
```

上述代码说明了数据库连接字串的基本格式。如果需要连接到本地的 mytable 数据库,则编写相应的 SQL 连接字串进行数据库的连接,示例代码如下所示。

```
string strcon;
strcon="server='(local)';database='mytable';uid='sa';pwd='sa';";
```

上述代码声明了一个数据库连接字串,SqlConnection 类将会通过此字串来进行数据库的连接。其中,server 是 SQL 服务器的地址,如果相对于应用程序而言数据库服务器是本地的,则只需要配置为(local)即可;而如果是远程服务器,则需要填写具体的 ip 地址。另外,uid 是数据库登录时的用户名,pwd 是数据库登录时使用的密码。在声明了数据库连接字串后,可以使用 SqlConnection 类进行连接,示例代码如下。

```
string strcon;                                                    //声明连接字串
strcon="server='(local)';database='mytable';uid='sa';pwd='sa';";//编写连接字串
SqlConnection con=new SqlConnection(strcon);                      //新建 SQL 连接
try
{
    con.Open();                                                   //打开 SQL 连接
    Label1.Text="连接数据库成功";                                 //提示成功信息
}
catch
{
    Label1.Text="无法连接数据库";                                 //提示失败信息
}
```

上述代码连接了本地数据库服务器中的 mytable 数据库,如果连接成功,则提示“连接数据库成功”;出现异常时,则提示“无法连接数据库”。

注意:*在使用 SqlConnection 类时,需要使用命名空间“using System. Data. SqlClient”;而连接 Access 数据库时,需要使用命名空间“using System. Data. OleDb”。*

(2) 填充 DataSet 数据集

DataSet 数据集表示来自一个或多个数据源数据的本地副本,是数据的集合,也可以看

作是一个虚拟的表。DataSet 对象允许 Web 窗体半独立于数据源运行。DataSet 能够提高程序的性能,因为 DataSet 从数据源中加载数据后,就会断开与数据源的连接,开发人员可以直接使用和处理这些数据,当数据发生变化并要更新时,则可以使用 DataAdapter 重新连接并更新数据源。DataAdapter 可以进行数据集的填充,创建 DataAdapter 对象的代码如下所示。

```
SqlDataAdapter da=new SqlDataAdapter("select * from news",con);
```

上述代码创建了一个 DataAdapter 对象并初始化 DataAdapter 对象,DataAdapter 对象的构造函数允许传递两个参数初始化,第一个参数为 SQL 查询语句,第二个参数为数据库连接的 SqlConnection 对象。初始化 DataAdapter 后,就需要将返回的数据的集合存放到数据集中,示例代码如下。

```
DataSet ds=new DataSet();          //创建数据集
da.Fill(ds, "tablename");          //Fill方法用于进行数据的填充
```

上述代码创建了一个 DataSet 对象并初始化 DataSet 对象,通过 DataAdapter 对象的 Fill 方法,可以将返回的数据存放到数据集 DataSet 中。DataSet 可以被看作是一个虚拟的表或表的集合,这个表的名称在 Fill 方法中被命名为 tablename。

(3) 显示 DataSet

当返回的数据被存放到数据集中后,可以通过循环语句遍历和显示数据集中的信息。当需要显示表中某一行字段的值时,可以通过 DataSet 对象获取相应行的某一列的值,示例代码如下。

```
ds.Tables["tablename"].Rows[0]["title"].ToString();
```

上述代码从 DataSet 对象中的虚表 tablename 的第 0 行中获取 title 列的值。当需要遍历 DataSet 时,可以使用 DataSet 对象中的 Count 来获取行数。DataSet 不仅可以通过编程的方法来实现显示,也可以使用 ASP. NET 中提供的控件来绑定数据集并显示。ASP. NET中提供了常用的显示 DataSet 数据集的控件,包括 Repeater、DataList、GridView 等数据绑定控件。将 DataSet 数据集绑定到 DataList 控件中可以方便地在控件中显示数据库中的数据并实现分页操作,示例代码如下所示。

```
DataList1.DataSource=ds;                //绑定数据集
DataList1.DataMember="tablename";
DataList1.DataBind();                   //绑定数据
```

上述代码能够将数据集 ds 中的数据绑定到 DataList 控件中。DataList 控件还能够实现分页、自定义模板等操作,非常便于开发人员对数据进行开发。从中可以看出,在 ADO. NET 中对数据库的操作基本上需要三个步骤,即创建一个连接、执行命令对象并显示,最后再关闭连接。使用 ADO. NET 的对象,不仅能够通过控件绑定数据源,也可以通过程序实现数据源的访问。ADO. NET 的过程如图 3-6 所示。

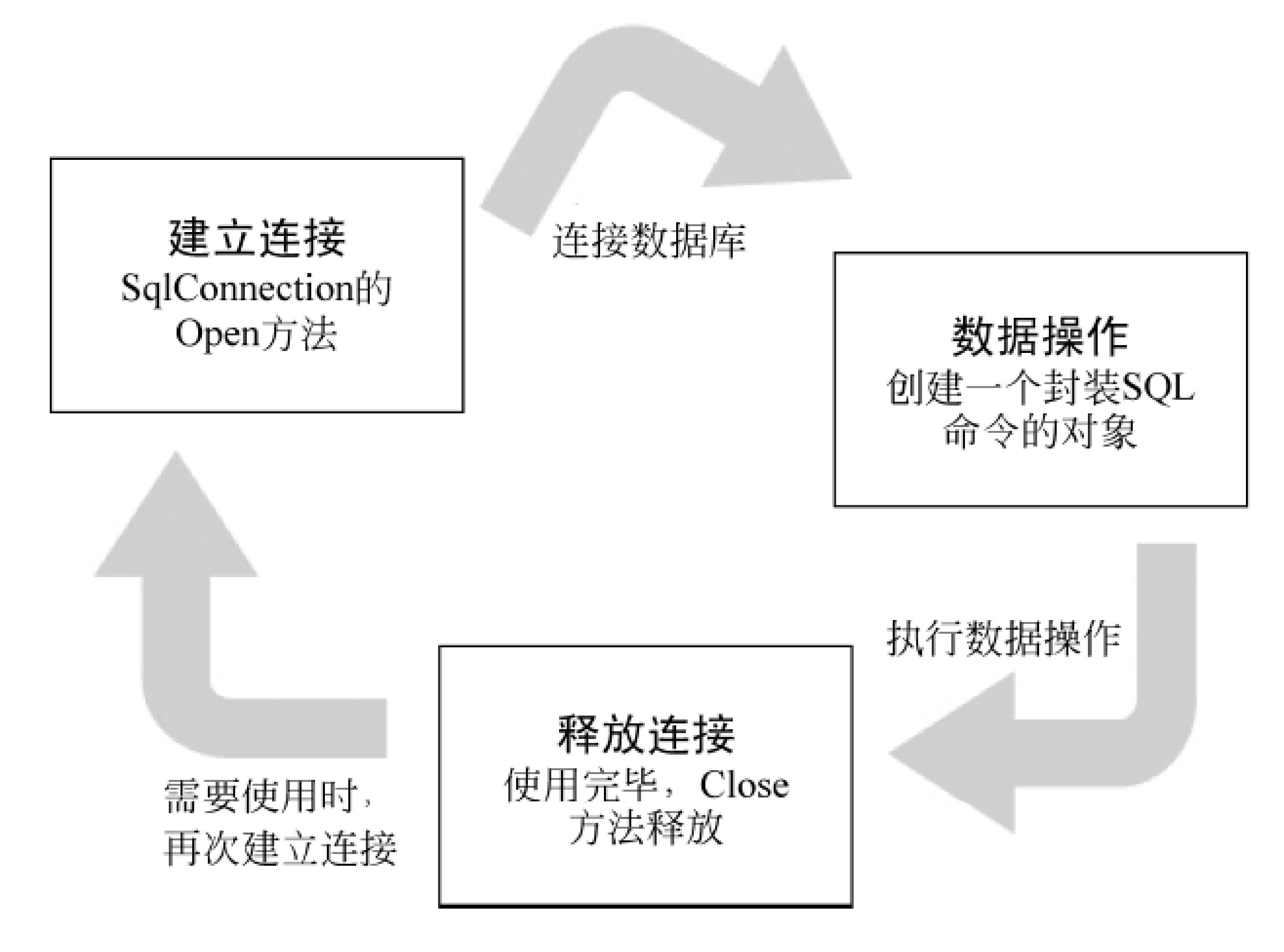


图 3-6　ADO. NET 规范的步骤

从图 3-6 中可以归纳出 ADO. NET 规范的步骤如下。

① 创建一个连接对象。

② 使用对象的 Open 方法打开连接。

③ 创建一个封装 SQL 命令的对象。

④ 调用执行命令的对象。

⑤ 执行数据库操作。

⑥ 执行完毕，释放连接。

掌握了这些初步的知识，就能够使用 ADO. NET 进行数据库的开发。

2. ADO. NET 常用对象

ADO. NET 提供了一些常用对象来方便开发人员进行数据库的操作，这些常用的对象通常会用在应用程序开发中，对于中级的开发人员而言，熟练地掌握这些常用的 ADO. NET 对象，能够自行封装数据库操作类，来简化开发。ADO. NET 的常用对象包括以下几种。

- Connection 对象。
- DataAdapter 对象。
- Command 对象。
- DataSet 对象。
- DataReader 对象。

上面的对象在. NET 应用程序操作数据中是非常重要的，它们不仅提供了数据操作的便利，同时，还提供了高级的功能给开发人员，为开发人员解决特定的需求提供帮助。

(1) Connection 连接对象

在. NET 开发中，通常情况下开发人员被推荐使用 Access 或者 SQL 作为数据源，若需要连接 Access 数据库，可以使用 System. Data. Oledb. OleDbConnection 对象来连接；若需要连接 SQL 数据库，则可以使用 System. Data. SqlClient. SqlConnection 对象来连接。本节主要讨论连接 Access 和 SQL 数据库。如需要连接 SQL 数据库，则需要使用命名空间 System. Data. SqlClient 和 System. Data. OleDb。示例代码如下。

```
using System.Data.SqlClient;              //使用 SQL 命名空间
using System.Data.Oledb                   //使用 Oledb 命名空间
```

① 使用 System. Data. SqlClient 连接 SQL Server。要连接到 SQL 数据库,则需要创建 SqlConnection 对象,代码如下。

```
SqlConnection con=new SqlConnection();    //创建连接对象
con.ConnectionString="server='(local)';database='mytable';uid='sa';pwd='sa'";
                                          //设置连接字串
```

上述代码创建了一个 SqlConnection 对象,并且配置了连接字串。SqlConnection 对象专门定义了一个专门接收连接字符串的变量 ConnectionString。当配置了 ConnectionString 变量后,就可以使用 Open()方法来打开数据库连接,示例代码如下。

```
SqlConnection con=new SqlConnection();
//创建连接对象
con.ConnectionString="server='(local)';database='mytable';uid='sa';pwd='sa'";try
{
        con.Open();
        Label1.Text="连接成功";
        con.Close();
}
catch
{
        Label1.Text="连接失败";
}
```

上述代码尝试判断是否打开数据库连接,使用 Open 方法能够建立应用程序与数据库之间的连接。与之相同的是,可以使用默认的构造函数来对数据库连接对象进行初始化,示例代码如下。

```
string str="server='(local)';database='mytable';uid='sa';pwd='Sa'";
//设置连接字符串
SqlConnection con=new SqlConnection(str);
//默认的构造函数
```

上述代码与使用 ConnectionString 变量的方法等价,其默认的构造函数中已经为 ConnectionString 变量进行了初始化。

② 使用 System. Data. OleDb 连接 Access 数据库。Access 是一种桌面级数据库,虽然与 SQL Server 相比,Access 数据库的性能和功能并不强大,但是 Access 却是最常用的数据库之一。对于小型应用和小型企业来说,Access 数据库也是开发中小型软件的最佳选择。Access 是 Office 组件之一,当安装了 Office 后,就可以新建 Access 数据库,在桌面或任何文件夹中右击就能够创建 Access 数据库。创建完成后,双击数据库文件就能够打开数据库并建立表和字段,如图 3-7 所示。

同样,Access 数据库也需要创建表和字段,基本方法与 SQL Server 数据库相同,但是在数据类型上,自动增长编号作为单独的数据类型而存在。开发人员能够在表窗口中创建表 mytable 和相应字段,如图 3-8 所示。

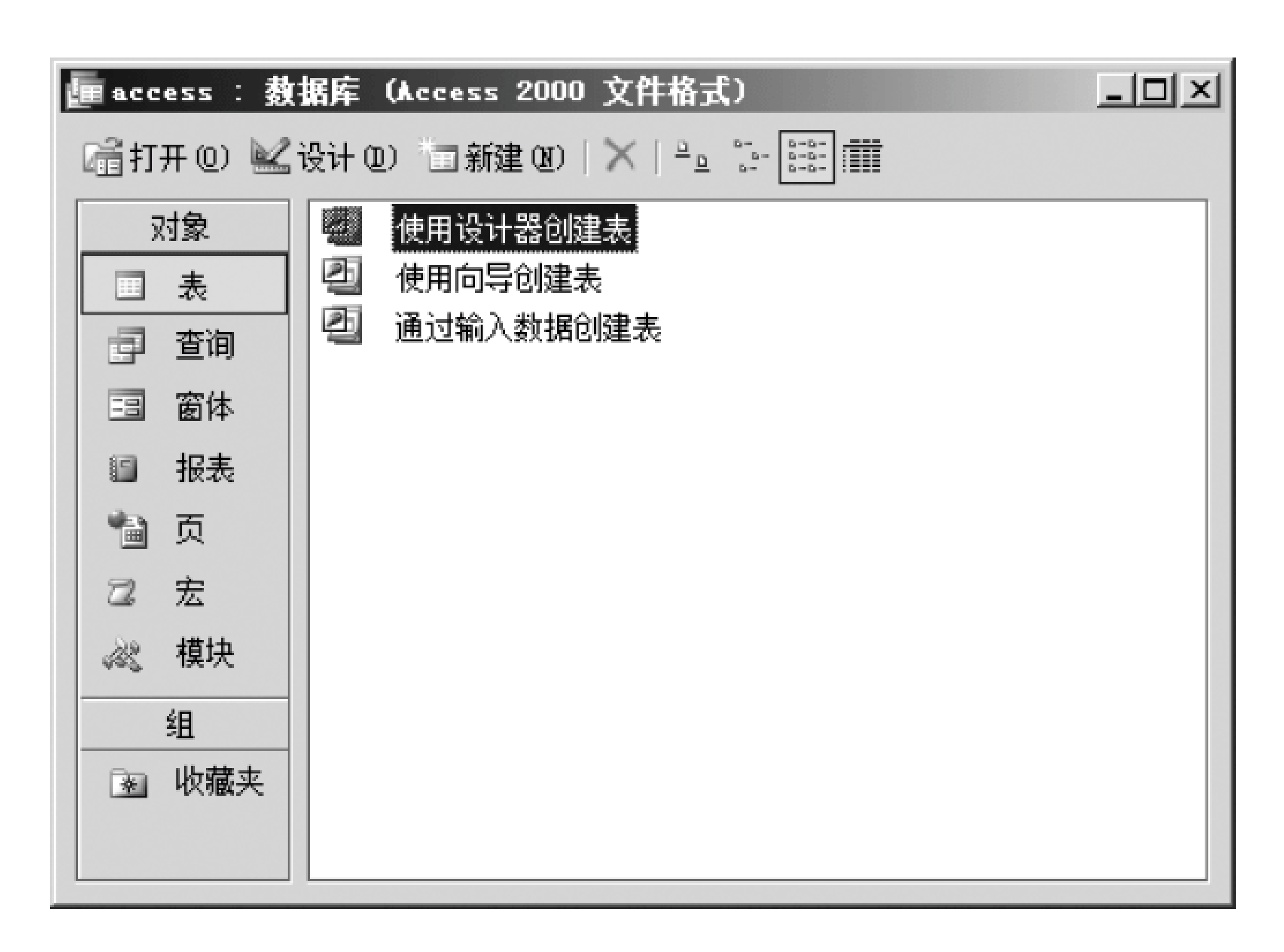


图 3-7　创建 Access 数据库



图 3-8　创建 Access 数据库的表

创建完成后可以使用 System. Data. OleDb 的对象进行数据库的连接和数据操作。注意，Access 数据库是一个桌面级的数据库，其数据都会存放在一个文件中而不是存放在数据库服务器中。在使用 System. Data. OleDb 时，只需要修改连接字符串即可。在这里需要强调一点的是，Access 数据库是一种桌面级的数据库，与文件类型的数据库类似，所以连接 Access 数据库时，必须指定数据库文件的路径，或者使用 Server. MapPath 来确定数据库文件的相对位置。示例代码如下所示。

```
string str="provider=Microsoft.Jet.OLEDB.4.0;Data Source="
                    +Server.MapPath("access.mdb")+"";
OleDbConnection con=new OleDbConnection(str);
try
{
    con.Open();
    Label1.Text="连接成功";
    con.Close();
```

```
}
catch(Exception ee)
{
    Label1.Text="连接失败";
}
```

Server.MapPath 能够确定文件相对于当前目录的路径，如果不使用 Server.MapPath，则需要指定文件在计算机的路径，如“D:\服务器\文件夹\数据库路径”。但是这样会暴露数据库的物理路径，让程序长期处于不安全的状态。无论是使用 System.Data.SqlClient 还是 System.Data.OleDb 创建数据库连接对象，都可以使用 Open 方法来打开连接。同样，也可以使用 Close 方法来关闭连接，示例代码如下所示。

注意：如果使用了连接池，虽然显式地关闭了连接对象，其实并不会真正地关闭与数据库之间的连接，这样能够保证再次进行连接时的连接性能。

```
SqlConnection con=new SqlConnection(str);
OleDbConnection con2=new OleDbConnection(str2);
//创建连接对象
con.Open();
//打开连接
con.Close();
//关闭连接
con2.Open();
//打开连接
con2.Close();
//关闭连接
```

(2) DataAdapter 适配器对象

在创建了数据库连接后，就需要对数据集 DataSet 进行填充，在这里就需要使用 DataAdapter 对象。在没有数据源时，DataSet 对象对保存在 Web 窗体可访问的本地数据库是非常实用的，这样降低了应用程序和数据库之间的通信次数。然而 DataSet 必须要与一个或多个数据源进行交互，DataAdapter 就提供 DataSet 对象和数据源之间的连接。为了实现这种交互，微软提供了 SqlDataAdapter 类和 OleDbDataAdapter 类。SqlDataAdapter 类和 OleDbDataAdapter 类各自适用情况如下。

- SqlDataAdapter：该类专用于 SQL 数据库，在 SQL 数据库中使用该类能够提高性能，SqlDataAdapter 与 OleDbDataAdapter 相比，无需使用 OLEDB 提供程序层，可直接在 SQL Server 上使用。
- OleDbDataAdapter：该类适用于由 OLEDB 数据提供程序公开的任何数据源，包括 SQL 数据库和 Access 数据库。

若要使一个使用 DataAdapter 对象的 DataSet 能够与一个数据源之间交换数据，则可以使用 DataAdapter 属性来指定需要执行的操作，这个属性可以是一条 SQL 语句或者是存储过程，示例代码如下所示。

```
string str="server='(local)';database='mytable';uid='sa';pwd='sa'";
//创建连接字符串
SqlConnection con=new SqlConnection(str);
con.Open();
```

```
//打开连接
SqlDataAdapter da=new SqlDataAdapter("select * from news", con);
//DataAdapter 对象
con.Close();
//关闭连接
```

上述代码创建了一个 DataAdapter 对象，DataSet 对象可以使用该对象的 Fill 方法来填充数据集。

(3) Command 执行对象

Command 对象可以使用数据命令直接与数据源进行通信。例如，当需要执行一条插入语句，或者删除数据库中的某条数据的时候，就需要使用 Command 对象。Command 对象的属性包括了数据库在执行某个语句的所有必要的信息，这些信息如下所示。

- Name：Command 的程序化名称。
- Connection：对 Connection 对象的引用。
- CommandType：指定是使用 SQL 语句或存储过程，默认情况下是 SQL 语句。
- CommandTest：命令对象包含的 SQL 语句或存储过程名。
- Parameters：命令对象的参数。

通常情况下，Command 对象用于数据的操作，例如执行数据的插入和删除，也可以执行数据库结构的更改，包括表和数据库。示例代码如下所示。

```
string str="server='(local)';database='mytable';uid='sa';pwd='sa'";
//创建数据库连接字符串
SqlConnection con=new SqlConnection(str);
con.Open();
//打开数据库连接
SqlCommand cmd=new SqlCommand("insert into news values ('title')",con);
//建立 Command 对象
```

上述代码使用了可用的构造函数并指定了查询字符串和 Connection 对象来初始化 Command 对象 cmd。通过指定 Command 对象的方法可以对数据执行具体的操作。

① ExecuteNonQuery 方法。当指定了一个 SQL 语句，就可以通过 ExecuteNonQuery 方法来执行语句的操作。ExecuteNonQuery 不仅可以执行 SQL 语句，开发人员也可以执行存储过程或数据定义语句来对数据库或目录执行构架操作。而使用 ExecuteNonQuery 时，ExecuteNonQuery 并不返回行，但是可以通过 Command 对象和 Parameters 进行参数传递。示例代码如下所示。

```
string str="server='(local)';database='mytable';uid='sa';pwd='sa'";
//创建数据库连接字符串
SqlConnection con=new SqlConnection(str);
con.Open();
SqlCommand cmd=new SqlCommand("insert into news values ('title')",con);
cmd.ExecuteNonQuery();
//执行 SQL 语句
```

运行上述代码后，会执行“INSERT INTO news VALUES('title')”这条 SQL 语句并向数据库中插入数据。值得注意的是，修改数据库的 SQL 语句，例如常用的 INSERT、

UPDATE 以及 DELETE 并不返回行。同样,很多存储过程同样不返回任何行。当执行这些不返回任何行的语句或存储过程时,可以使用 ExecuteNonQuery。但是 ExecuteNonQuery 语句也会返回一个整数,表示受已执行的 SQL 语句或存储过程影响的行数,示例代码如下所示。

```
string str="server='(local)';database='mytable';uid='sa';pwd='sa'";
SqlConnection con=new SqlConnection(str);
//创建连接对象
con.Open();
//打开连接
SqlCommand cmd=new SqlCommand("delete from mynews", con);
//构造 Command 对象
Response.Write("该操作影响了('+cmd.ExecuteNonQuery().ToString()+')行");
//执行 SQL 语句
```

上述代码执行了语句"DELETE FROM mynews",并将影响的行数输出到字符串中。开发人员能够使用 ExecuteNonQuery 语句进行数据库操作和数据库操作所影响行数的统计。

② ExecuteNonQuery 执行存储过程。ExecuteNonQuery 不仅能够执行 SQL 语句,同样可以执行存储过程和数据定义语言来对数据库或目录执行构架操作(如 CREATE TABLE 等)。在执行存储过程之前,必须先创建一个存储过程,然后在 SqlCommand 方法中使用存储过程。在 SQL Server 管理器中可以新建查询创建存储过程,示例代码如下所示。

```
CREATE PROC getdetail
(
    @id int,
    @title varchar(50) OUTPUT
)
AS
SET NOCOUNT ON
DECLARE @newscount int
SELECT @title=mynews.title,@newscount=COUNT(mynews.id)
FROM mynews
WHERE(id=@id)
GROUP BY mynews.title
RETURN @newscount
```

上述存储过程返回了数据库中新闻的标题内容。"@id"表示新闻的 id,"@title"表示新闻的标题,此存储过程将返回"@title"的值,并且返回新闻的总数。创建存储过程后,就可以使用 SqlParameter 调用命令对象 Parameters 参数的集合的 Add 方法进行参数传递,并指定相应的参数,示例代码如下所示。

```
string str="server='(local)';database='mytable';uid='sa';pwd='Sa'";
SqlConnection con=new SqlConnection(str);
con.Open();                                         //打开连接
SqlCommand cmd=new SqlCommand("getdetail", con);    //使用存储过程
cmd.CommandType=CommandType.StoredProcedure;              //设置 Command 对象的类型
```

```
SqlParameter spr;                                              //表示执行一个存储过程
spr=cmd.Parameters.Add("@id",SqlDbType.Int);                   //增加参数 id
spr=cmd.Parameters.Add("@title",SqlDbType.NChar,50);           //增加参数 title
spr.Direction=ParameterDirection.Output;                       //该参数是输出参数
spr=cmd.Parameters.Add("@count",SqlDbType.Int);                //增加 count 参数
spr.Direction=ParameterDirection.ReturnValue;                  //该参数是返回值
cmd.Parameters["@id"].Value=1;                                 //为参数初始化
cmd.Parameters["@title"].Value=null;                           //为参数初始化
cmd.ExecuteNonQuery();                                         //执行存储过程
Label1.Text=cmd.Parameters["@count"].Value.ToString();         //获取返回值
```

上述代码使用了现有的存储过程，并为存储过程传递了参数，当参数被存储过程接受并运行后，会返回一个存储过程中指定的返回值。当执行完毕后，开发人员可以通过cmd.Parameters来获取其中一个变量的值。

③ ExecuteScalar 方法。Command 的 Execute 方法提供了返回单个值的功能。在很多时候，开发人员需要获取刚刚插入的数据的 ID 值，或者可能需要返回 Count(*)、Sum(Money)等聚合函数的结果，则可以使用 ExecuteScalar 方法。示例代码如下所示。

```
string str="server='(local)';database='mytable';uid='sa';pwd='sa'";
//设置连接字符串
SqlConnection con=new SqlConnection(str);
//创建连接
con.Open();
//打开连接
SqlCommand cmd=new SqlCommand("select count(*)from mynews", con);
//创建 Command
Label1.Text=cmd.ExecuteScalar().ToString();
//使用 ExecuteScalar 执行
```

上述代码创建了一个连接，并创建了一个 Command 对象，使用了可用的构造函数来初始化对象。当使用 ExecuteScalar 执行方法时，会返回单个值。ExecuteScalar 方法同样可以执行 SQL 语句，但是与 ExecuteNonQuery 方法不同的是，当语句不为 SELECT 时，则返回一个没有任何数据的 System.Data.SqlClient.SqlDataReader 类型的集合。ExecuteScalar 方法通常是用来执行具有返回值的 SQL 语句，例如上面所说的当插入一条新数据时，返回刚插入的数值的 ID 号。这种功能在自动增长类型的数据库设计中经常被用到，示例代码如下所示。

```
string str="server='(local)';database='mytable';uid='sa';pwd='sa'";
                                                               //设置连接字符串
SqlConnection con=new SqlConnection(str);                      //创建连接
con.Open();                                                    //打开连接
SqlCommand cmd=new SqlCommand("insert into mynews values
    ('this is a new title!')SELECT @@IDENTITY as 'bh'", con);  //插入新数据
Label2.Text=cmd.ExecuteScalar().ToString();                    //获取返回的 ID值
```

上述代码使用了“SELECT @@IDENTITY as”语法，“SELECT @@IDENTITY”语法会自动获取刚插入的自动增长类型的值。例如，当表中有 100 条数据时，插入一条数据后

数据量就成 101，为了不需要再次查询就获得 101 这个值，则可以使用“SELECT @@IDENTITY as”语法。当使用了“SELECT @@IDENTITY as”语法进行数据操作时，ExecuteScalar 方法会返回刚插入的数据的 ID，这样就无需再次查询刚插入的数据。

(4) DataSet 数据集对象

DataSet 是 ADO.NET 中的核心概念，作为初学者，可以把 DataSet 想象成虚拟表，但是这个表不能用简单的表来表示，这个表可以想象成具有数据库结构的表，并且这个表是存放在内存中的。由于 ADO.NET 中 DataSet 的存在，开发人员能够屏蔽数据库与数据库之间的差异，从而获得一致的编程模型。DataSet 能够支持多表、表间关系、数据库约束等，可以模拟一个简单的数据库模型。DataSet 对象模型如图 3-9 所示。

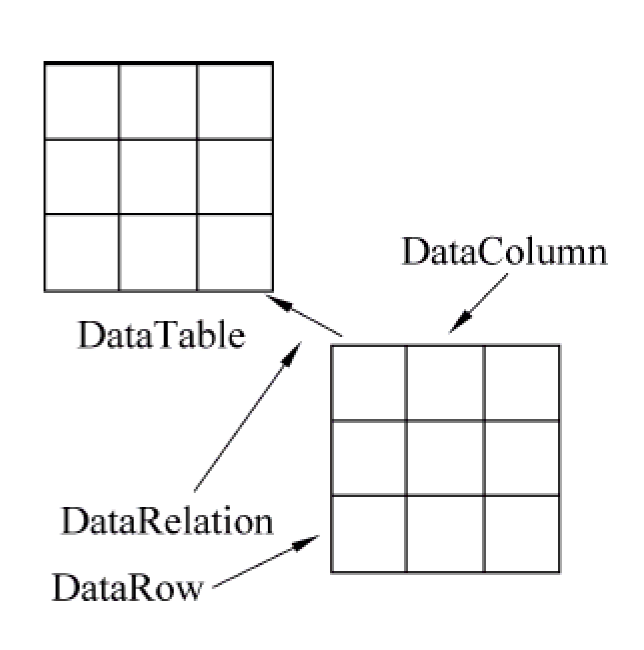


图 3-9　DataSet 对象模型

图 3-9 简要地介绍了常用对象之间的构架关系。在 DataSet 中，主要包括 TablesCollection、RelationsCollection、ExtendedProperties 几个重要对象。

① TablesCollection 对象。在 DataSet 中，表的概念是用 DataTable 来表示的。DataTable 在 System.Data 中定义，它能够表示存储在内存中的一张表。它包含一个 ColumnsCollection 的对象，代表数据表的各个列的定义。同时，它也包含了 RowsCollection 对象，这个对象包含 DataTable 中的所有数据。

② RelationsCollection 对象。在各个 DataTable 对象之间，是通过使用 RelationsCollection 来表达各个 DataTable 对象之间的关系的。RelationsCollection 对象可以模拟数据库中的约束关系。例如当一个包含外键的表被更新时，如果不满足主键—外键约束，这个更新操作就会失败，系统会抛出异常。

③ DataTable 数据表对象。DataTable 是 DataSet 中的常用的对象，它和数据库中表的概念十分相似。开发人员可以将 DataTable 想象成一个表，并且可以通过编程的方式创建一个 DataTable 表。示例代码如下所示。

```
DataTable Table=new DataTable("mytable");
//创建一个 DataTable对象
Table.CaseSensitive=false;
//设置不区分大小写
Table.MinimumCapacity=100;
//设置 DataTable初始的大小
Table.TableName="newtable";
//设置 DataTable的名称
```

上述代码创建了一个 DataTable 对象，并为 DataTable 对象设置了若干属性，这些属性都是常用的属性，其作用分别说明如下。

- CaseSensitive：此属性设置表中的字符串是否需要区分大小写。若无特殊情况，一般设置为 False。该属性对查找、排序、过滤等操作有很大的影响。
- MinimumCapacity：设置创建的数据表的最小记录空间。

• TableName：指定数据表的名称。

一个表必须有一个列，而 DataTable 必须包含列。当创建了一个 DataTable 后，就必须向 DataTable 中增加列。表中列的集合形成了二维表的数据结构。开发人员可以使用 Columns 集合的 Add 方法向 DataTable 中增加列，Add 方法带有两个参数，一个是表列的名称，一个是该列的数据类型。示例代码如下所示。

```
DataTable Table=new DataTable("mytable");
//创建一个 DataTable
Table.CaseSensitive=false;
//设置不区分大小写
Table.MinimumCapacity=100;
//设置 DataTable 初始的大小
Table.TableName="newtable";
//设置 DataTable 的名称
DataColumn Colum=new DataColumn();
//创建一个 DataColumn
Colum=Table.Columns.Add("id", typeof(int));
//增加一个列
Colum=Table.Columns.Add("title", typeof(string));
//增加一个列
```

上述代码创建了一个 DataTable 和一个 DataColumn 对象，并通过 DataTable 的 Columns. Add 方法增加 DataTable 的列。

注意：上述代码中，DataTable 列的数据类型使用的只能是. NET 中的数据类型，因为其并不是真实的数据库，所以不能直接使用数据库类型，必须使用 typeof 方法把. NET 中的数据类型转换成数据库类型。

④ DataRow 数据行对象。在创建了表和表中列的集合，并使用约束定义表的结构后，可以使用 DataRow 对象向表中添加新的数据库行，这一操作同数据库中的 INSERT 语句的概念类似。插入一个新行，首先要声明一个 DataRow 类型的变量。使用 NewRow 方法能够返回一个新的 DataRow 对象。DataTable 会根据 DataColumnCollection 定义的表的结构来创建 DataRow 对象。示例代码如下所示。

```
DataRow Row=Table.NewRow();
//使用 DataTable 的 NewRow 方法创建一个新 DataRow 对象
```

上述代码使用 DataTable 的 NewRow 方法创建了一个新 DataRow 对象，当使用该对象添加了新行之后，必须使用索引或者列名来操作新行，示例代码如下所示。

```
Row[0]=1;
//使用索引赋值列
Row[1]="datarow";
//使用索引赋值列
```

上述代码通过索引来为一行中各个列赋值。从数组的语法可以知道，索引都是从第 0 个位置开始。将 DataTable 想象成一个表，从左到右从 0 开始索引，直到数值等于列数减 1 为止。为了提高代码的可读性，也可以通过直接使用列名来添加新行，示例代码如下所示。

```
Row["bh"]=1;
//使用列名赋值列
Row["title"]="datarow";
//使用列名赋值列
```

通过直接使用列名来添加新行与使用索引添加新行的效果相同，通过使用列名能够让代码更加可读，便于理解，但是也暴露了一些机密内容（如列值）。在把数据插入到新行后，使用 Add 方法将该行添加到 DataRowCollection 中。

（5）DataReader 数据访问对象

DataSet 的最大好处在于，能够提供无连接的数据库副本，DataSet 对象在表的生命周期内会为这些表进行内存的分配和维护。如果有多个用户同时对一台计算机进行操作，内存的使用就会变得非常紧张。当需要对数据进行一些简单的操作时，就无需保持 DataSet 对象的生命周期，可以使用 DataReader 对象。当使用 DataReader 对象时，不会像 DataSet 那样提供无连接的数据库副本。DataReader 类被设计为产生只读、只进的数据流。这些数据流都是从数据库返回的。所以，每次的访问或操作只有一个记录保存在服务器的内存中。相对于 DataSet 而言，DataReader 具有较快的访问能力，并且能够使用较少的服务器资源，DataReader 具有以下快速的数据库访问、只进和只读、减少服务器资源等特色。DataReader 类是轻量级的，相比之下 DataReader 对象的速度要比 DataSet 要快。因为 DataSet 在创建和初始化时，可能是一个或多个表的集合，并且 DataSet 具有向前、向后读写和浏览的能力，所以当创建一个 DataSet 对象时，会造成额外的开销。当对数据库的操作没有太大的要求时，可以使用 DataReader 显示数据。这些数据可以与单个 list-bound 控件绑定，也可以填充 List 接口。当不需要复杂的数据库处理时，DataReader 能够较快地完成数据的显示。因为 DataReader 并不是数据在内存中的表示形式，所以使用 DataReader 对服务器占用的资源很少。DataReader 对象可以使用 Read 方法来进行数据库遍历，当使用 Read 方法时，可以以编程的方式自定义数据库中数据的显示方式。当开发自定义控件时，可以将这些数据整合到 HTML 中，并显示数据。

DataAdapter 对象能够自动地打开和关闭连接，而 DataReader 对象需要用户手动来管理连接。DataReader 对象和 DataAdapter 对象很相似，都可以从 SQL 语句和一个连接中初始化。创建 DataReader 对象，需要创建一个 SqlCommand 对象来代替 SqlDataAdapter 对象。与 SqlDataAdapter 对象类似的是，DataReader 可以从 SQL 语句和连接中创建 Command 对象。创建对象后，必须显式地打开 Connection 对象。示例代码如下所示。

```
string str="server='(local)';database='mytable';uid='sa';pwd='sa'";
SqlConnection con=new SqlConnection(str);
con.Open();
//打开连接
SqlCommand cmd=new SqlCommand("select * from mynews", con);
//创建 Command 对象
SqlDataReader dr=cmd.ExecuteReader();
//创建 DataReader 对象
con.Close();
```

上述代码创建了一个 DataReader 对象，从上述代码中可以看出，创建 DataReader 对象

必须经过如下几个步骤。

- 创建和打开数据库连接。
- 创建一个 Command 对象。
- 从 Command 对象中创建 DataReader 对象。
- 调用 ExecuteReader 对象。

DataReader 对象的 Read 方法可以判断 DataReader 对象中的数据是否还有下一行，并将游标移到下一行。通过 Read 方法可以判断 DataReader 对象中的数据是否读完。示例代码如下所示。

```
while (dr.Read())
```

通过 Read 方法可以遍历读取数据库中行的信息。当读取到一行时，需要获取某列的值只需要使用“[”和“]”运算符来确定某一列的值即可，示例代码如下所示。

```
while (dr.Read())
{
    Response.Write(dr["title"].ToString()+"<hr/>");
}
```

上述代码通过 dr["title"]来获取数据库中 title 这一列的值，同样也可以通过索引来获取某一列的值，示例代码如下所示。

```
while (dr.Read())
{
    Response.Write(dr[1].ToString()+"<hr/>");
}
```

示例 2：假定可以连接到 Microsoft SQL Server 2008 或更高版本上的 AdventureWorks 示例数据库，请使用 SqlDataReader 从 HumanResources. Department 表中返回记录列表(如图 3-10 所示)。

① 拖动 gridview 控件到 default. aspx 页面，单击 gridview 智能提示，选择“自动套用格式”菜单项，单击“彩色型”列表项，再单击“确定”按钮。

② 按 F7 功能键切换到代码页面，在 Page_Load 事件中输入以下代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    String connectionString="Data Source=.;Initial Catalog=AdventureWorks;
        Integrated Security=True; MultipleActiveResultSets=true";
    SqlConnection con=new SqlConnection(connectionString);
    con.Open();
    SqlCommand cmd=new SqlCommand("select * from HumanResources.
    department", con);
    SqlDataReader sdr=cmd.ExecuteReader();
    GridView1.DataSource=sdr;
    GridView1.DataBind();
    con.Close();
}
```

DepartmentID	Name	GroupName	ModifiedDate
1	Engineering	Research and Development	1998/6/1 0:00:00
2	Tool Design	Research and Development	1998/6/1 0:00:00
3	Sales	Sales and Marketing	1998/6/1 0:00:00
4	Marketing	Sales and Marketing	1998/6/1 0:00:00
5	Purchasing	Inventory Management	1998/6/1 0:00:00
6	Research and Development	Research and Development	1998/6/1 0:00:00
7	Production	Manufacturing	1998/6/1 0:00:00
8	Production Control	Manufacturing	1998/6/1 0:00:00
9	Human Resources	Executive General and Administration	1998/6/1 0:00:00
10	Finance	Executive General and Administration	1998/6/1 0:00:00
11	Information Services	Executive General and Administration	1998/6/1 0:00:00
12	Document Control	Quality Assurance	1998/6/1 0:00:00
13	Quality Assurance	Quality Assurance	1998/6/1 0:00:00
14	Facilities and Maintenance	Executive General and Administration	1998/6/1 0:00:00
15	Shipping and Receiving	Inventory Management	1998/6/1 0:00:00
16	Executive	Executive General and Administration	1998/6/1 0:00:00

图 3-10　数据显示

③ 按 F5 功能键运行程序，出现如图 3-10 所示的页面。

3.4　项 目 实 施

3.4.1　任务 1：会员注册 UI 设计

1. 任务目标

(1) 能熟练操作 VS2012 集成开发环境。

(2) 能熟练使用 ASP. NET 常用服务器控件。

(3) 能使用 DIV+CSS 进行网页布局。

2. 任务内容

(1) 从 Layout. master 模板页创建内容页。

(2) 使用服务器控件设计注册页面界面。

3. 任务实施步骤

该任务首先从 Layout. master 母版页创建内容页，接着使用 DIV+CSS 进行页面布局，最后使用常用的服务器控件设计界面(如图 3-11 所示)。

① 右击 Shop 文件夹，选择“添加”→“添加新项”命令，打开“添加新项”对话框。

② 单击“Web 页面”列表项，在“名称”文本框中输入 reg. aspx，单击“选择母版页”，单击“添加”按钮，打开“选择模板页”对话框，单击 Layout. master 文件，再单击“确定”按钮。

③ 打开 mystyle. cs 文件，输入以下代码。

```
.reg_div
{
    width:600px;
    height:30px;
}
```

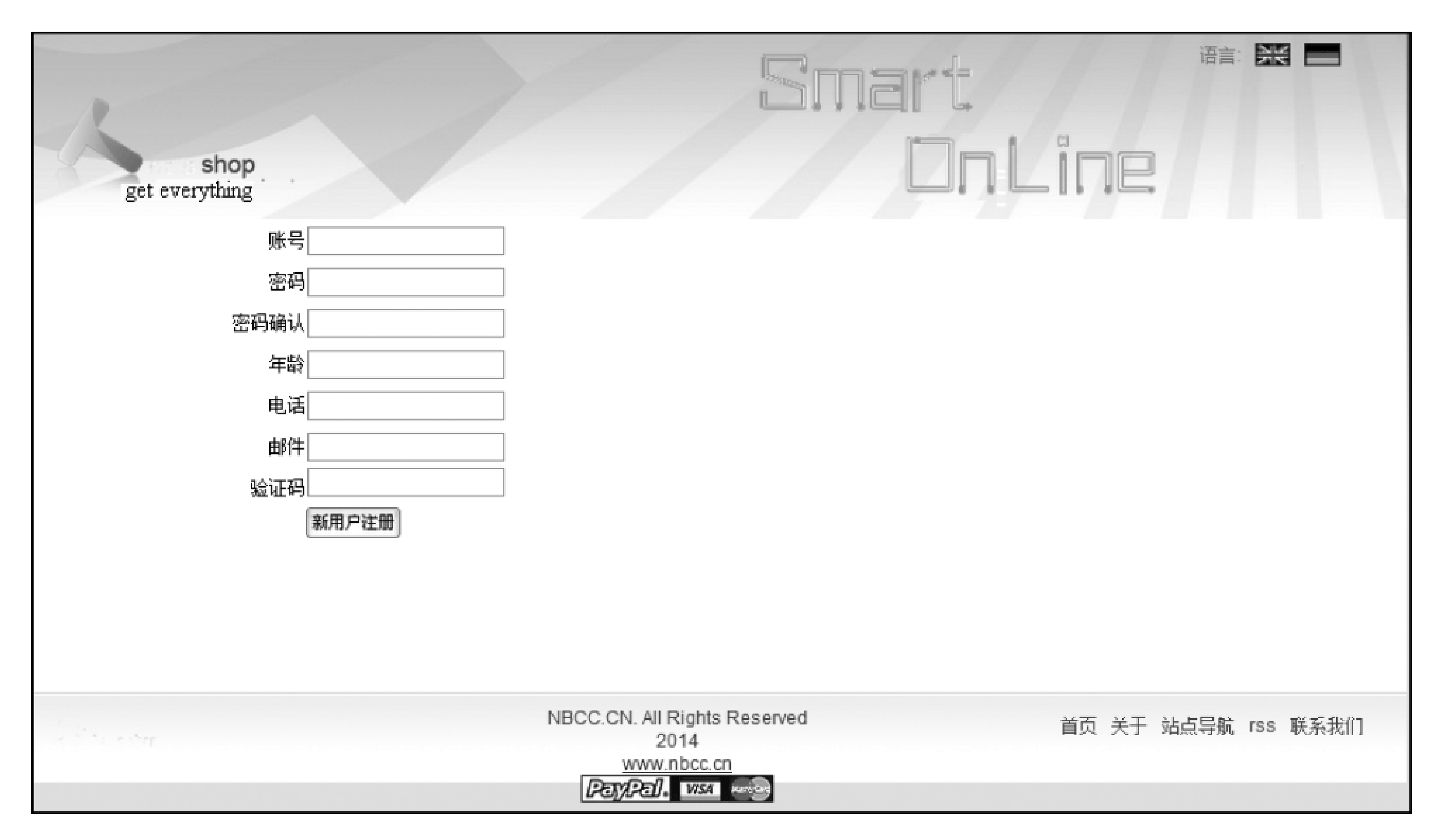


图 3-11　任务 1 效果图

```
.reg_div_tips
{
    float:left;
    width:200px;
    height:30px;
    line-height:30px;
    text-align:right;
}
.reg_div_content
{
    float:left;
    width:400px;
    height:30px;
    text-align:center;
    line-height:30px;
    text-align:left;
}
.reg_div_content2
{
    float:left;
    width:160px;
    height:30px;
    text-align:center;
    line-height:30px;
    text-align:left;
}
.reg_div_content3
{
    float:left;
    width:40px;
    height:30px;
    text-align:center;
```

```
    line-height:30px;
    text-align:left;
}
.reg_div_content4
{
    float:left;
    width:80px;
    height:30px;
    text-align:center;
    line-height:30px;
    text-align:left;
}

.img_disp
{
    width:30px;
    height:20px;
    line-height:1px;
}
```

④ 拖动 DIV 控件到页面中，分别更改各个 DIV 的样式。

```
<div class="reg_div">
    <div class="reg_div_tips">账号</div>
    <div class="reg_div_content"></div>
</div>
    <div class="reg_div">
    <div class="reg_div_tips">密码</div>
    <div class="reg_div_content"></div>
</div>
    <div class="reg_div">
    <div class="reg_div_tips">密码确认</div>
    <div class="reg_div_content"></div>
</div>
    <div class="reg_div">
    <div class="reg_div_tips">年龄</div>
    <div class="reg_div_content"></div>
</div>
    <div class="reg_div">
    <div class="reg_div_tips">电话</div>
    <div class="reg_div_content"></div>
</div>
    <div class="reg_div">
    <div class="reg_div_tips">邮件</div>
    <div class="reg_div_content"></div>
</div>
    <div class="reg_div">
    <div class="reg_div_tips">验证码</div>
    <div class="reg_div_content2"></div>
    <div class="reg_div_content3"></div>
    <div class="reg_div_content4"></div>
```

```
</div>
    <div class="reg_div">
    <div class="reg_div_tips" >
    </div><div class="reg_div_content">
</div>
```

⑤ 拖动 TextBox 控件到页面中。每行用一个 DIV 表示，它的样式类是 reg_div，其中包含两个 DIV，第二个 DIV 用于放置 TextBox 控件。

⑥ 右击账号文本框，选择"属性选项"命令，单击 ID 属性，为其输入 TxtAccount。以此方式更改各个文本框的 ID 属性。

3.4.2 任务 2：会员输入信息验证

1. 任务目标

(1) 能使用 ASP.NET 验证控件验证输入信息。

(2) 能生成图像型验证码。

2. 任务内容

(1) 能使用 ASP.NET 验证控件验证 Smart On Line 电子商城会员注册信息。

(2) 能使用验证码技术防止恶意注册。

3. 任务实施步骤

制作 Smart On Line 商城母版页(见图 3-12)首先是采用必填验证控件对所有必须输入的文本框进行验证；接着采用比较验证器验证"密码"和"密码确认"是否一致和范围验证控件验证年龄；然后采用正则表达式验证器对"电话"和"邮件"进行验证；下一步对输入的账号采用自定义验证控件验证输入账号是否存在；最后使用验证码技术检验验证码是否一致。



图 3-12 任务 2 效果图

① 对"账号"、"密码"、"密码确认"、"年龄"、"电话"和"邮件"文本框进行必填验证。拖动 RequiredFieldValidator 空间到相应文本框右侧。右击"账号"文本框，选择"属性"命令，打开"属性窗口"(如图 3-13 所示)，单击 ControlToValidate 属性，单击 TxtAccount(账号文本框的 ID)。接着单击 ErrorMessage 属性，输入"请输入账号"；单击 Display 属性，单击 Dynamic 列表项；然后单击 ForeColor 属性，输入＃990000。用类似操作方式设置其他 6 个

必填验证控件的属性。

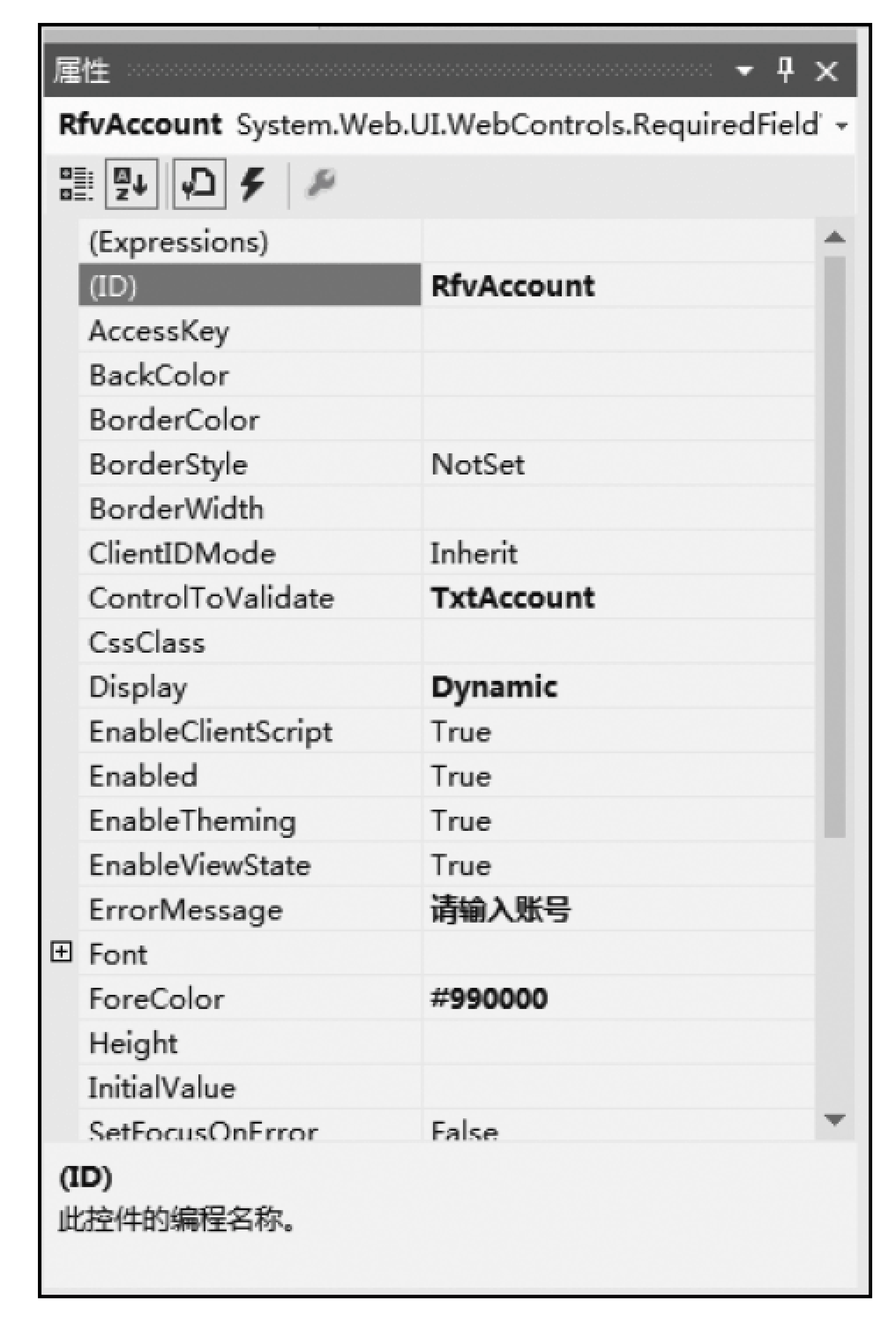


图 3-13 “属性”窗口

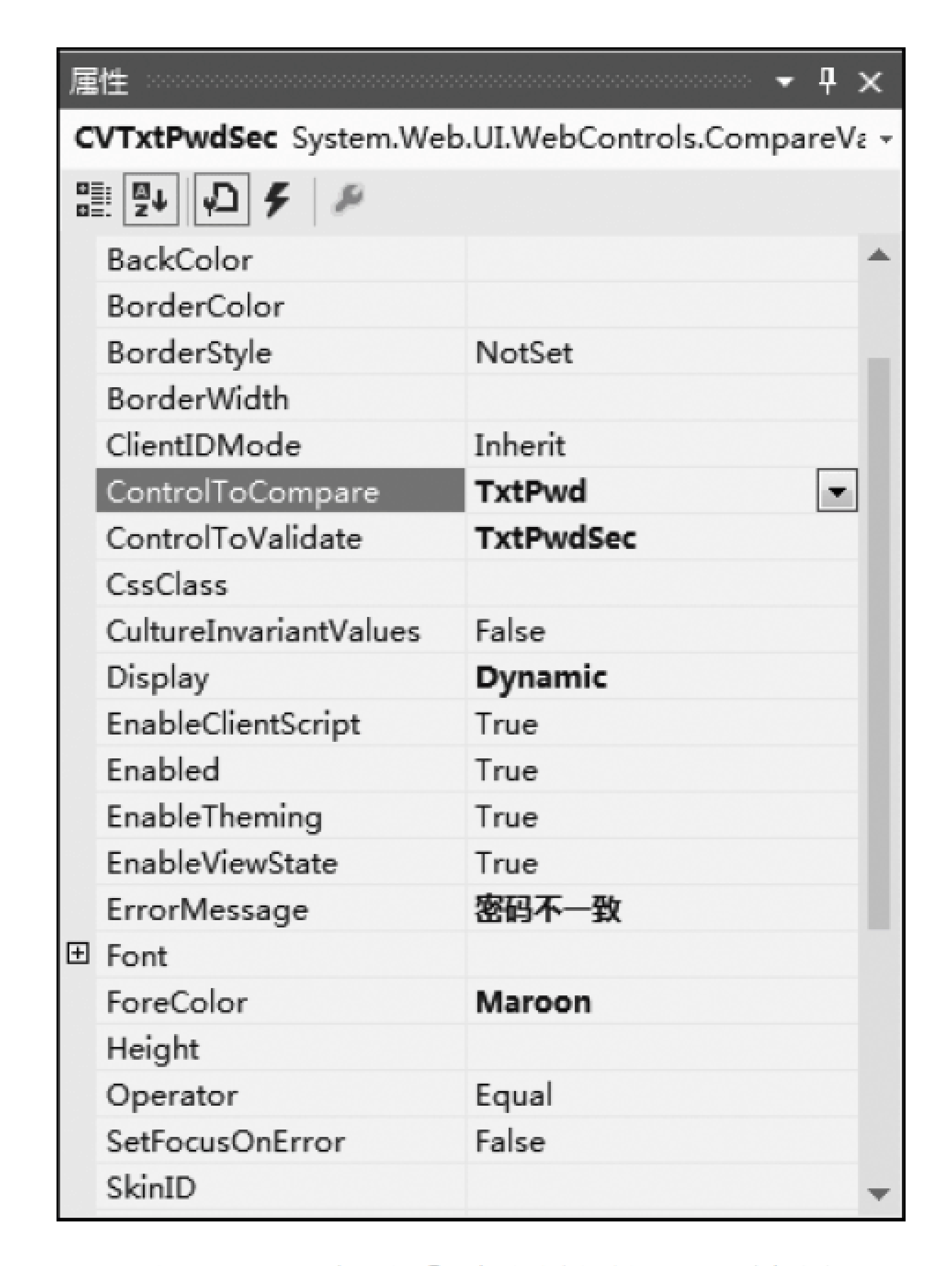


图 3-14 步骤②中属性的设置效果

② 拖动 CompareValidator 到“密码确认”文本框右侧，单击 ControlToValidator 属性，单击右边的下拉列表，单击 TxtPwdConfirm 选项。单击 ControlToCompare 属性，再单击右边的下拉列表，单击 TxtPwd 选项。单击 ErrorMessage 属性，输入“密码不一致”。单击 Display 属性，单击右边下拉列表，再单击 Dynamic 选项，其他的特别是 Type 属性保持不变，如图 3-14 所示。

③ 使用范围验证控件验证“年龄文本框”输入值。拖动 RangeValidator 到年龄文本框右侧，右击 RangeValidator1 验证控件，选择“属性”命令，打开“属性”窗口。单击 ControlToValiator 属性，单击右边的下拉列表，单击 TxtAge 选项；单击 Display 属性，单击 Dynamic 选项；单击 ErrorMessage 属性，输入“年龄必须大于 18 岁”；单击 MaxinumValue 属性，输入 200(人的年龄不超过 200 岁，所以这里设置最大值为 200)；单击 MininumValue 属性，输入 18。还需进一步单击 Type 属性，单击右边的下拉列表，再单击 Integer 选项，如图 3-15 所示。

④ 拖动 RegularExpressionValidator 到“电话”文本框右侧。右击 RegularExpressionValidator1 验证控件，选择“属性”命令，打开“属性”对话框。单击 ValidationExpress 属性，单击右边的按钮，打开“正则表达式编辑器”对话框，单击“中华人民共和国电话号码”选项，单击“确定”按钮(如图 3-16 所示)。单击 ErrorMessage 属性，输入“电话号码格式不一致”。单击 Display 属性，单击 Dynamic 选项。单击 ControlToValidator 属性，单击右边的下拉列表，再单击 TxtPhone 选项。

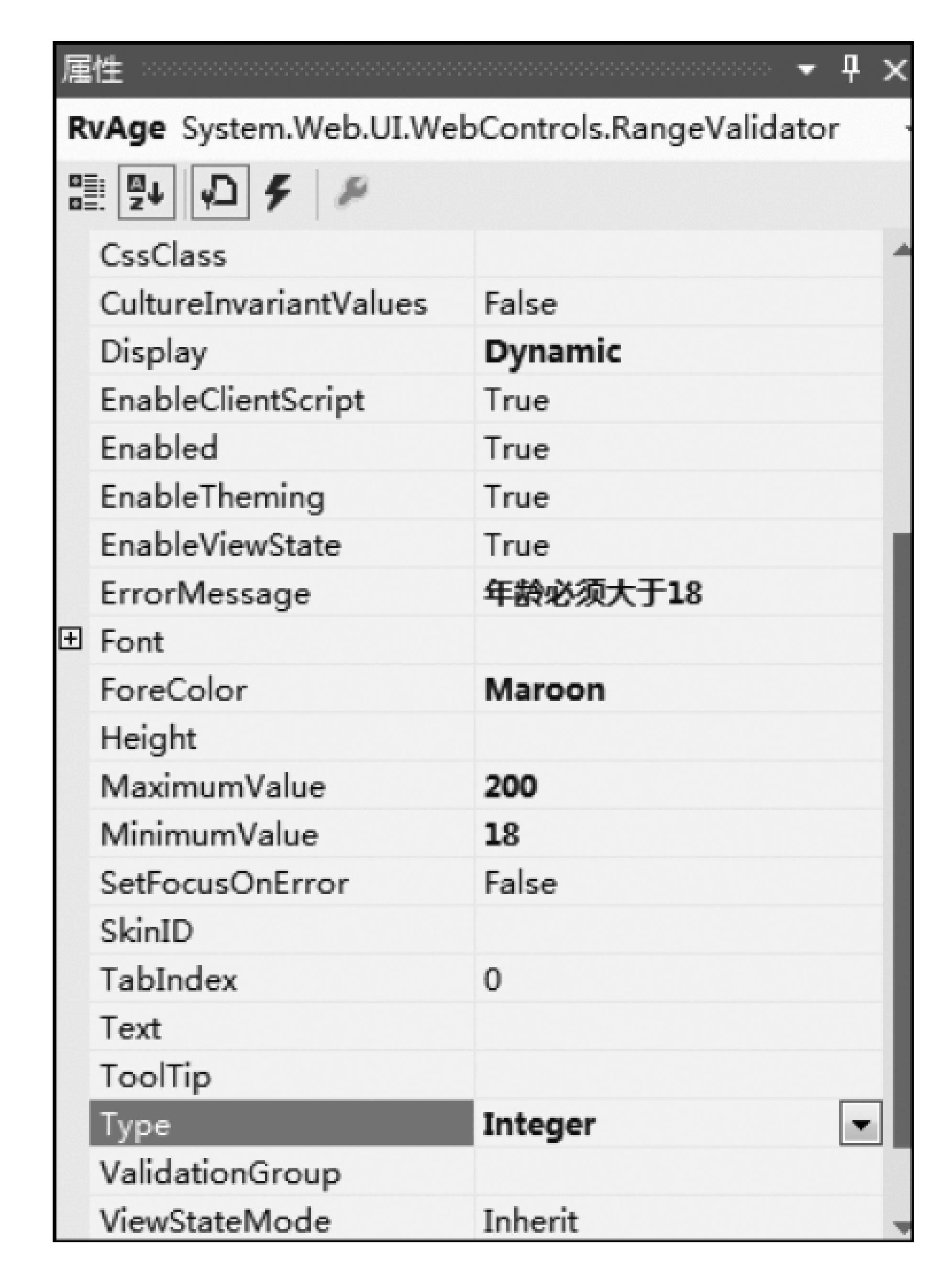


图 3-15　步骤③中属性的设置



图 3-16　“正则表达式编辑器”对话框

⑤ 拖动 RegularExpressionValidator 并放到“邮件”文本框右侧。右击 RegularExpressionValidator2 验证控件，选择“属性”命令，打开“属性”对话框。单击 ValidationExpress 属性，单击右边的按钮，打开“正则表达式编辑器”对话框，单击“Internet 电子邮件地址”选项，单击“确定”按钮。单击 ErrorMessage 属性，输入“电子邮件格式错误”。单击 Display 属性，单击 Dynamic 选项。单击 ControlToValidator 属性，单击右边的下拉列表，单击 TxtEmail 选项。

⑥ 生成图像型验证。右击 App_Code 文件夹，选择“添加”→“添加新项”命令，打开“添加新项”对话框，单击“类”列表项，在“名称”文本框中输入 CheckCode。双击 CheckCode. cs 文件，输入下列代码来产生随机数。

```
private string RndNum(int VcodeNum)
{
    string Vchar="0,1,2,3,4,5,6,7,8,9";
    string[] VcArray=Vchar.Split(',');
    string VNum="";
    int temp=-1;
    Random rand=new Random();
    for (int i=1; i<VcodeNum+1; i++)
    {
        if (temp ! =-1)
        {
            rand=new Random(i * temp * unchecked((int)DateTime.Now.Ticks));
        }
        int t=rand.Next(VcArray.Length);
        if (temp ! =-1 && temp==t)
        {
            return RndNum(VcodeNum);
```

```
        }
        temp=t;
        VNum+=VcArray[t];
    }
    return VNum;
}
```

⑦ 图像型验证码需要以图像格式显示验证码，要根据字符型验证码转换成图片，将其显示到浏览器中，然后将图片释放。代码如下。

```
private void checkCodes(string checkCode)
{
  int iwidth=(int)(checkCode.Length * 13);
  System.Drawing.Bitmap image=new System.Drawing.Bitmap(iwidth, 23);
  Graphics g=Graphics.FromImage(image);
  g.Clear(Color.White);
  //定义颜色
  Color[] c={ Color.Black, Color.Red, Color.DarkBlue, Color.Green,
      Color.Orange, Color.Brown, Color.DarkCyan, Color.Purple };
  //定义字体
  string[] font={ "Verdana", "Microsoft Sans Serif", "Comic Sans MS",
       "Arial", "宋体" };
  Random rand=new Random();
  //随机输出噪点
  for (int i=0; i<50; i++)
  {
      int x=rand.Next(image.Width);
      int y=rand.Next(image.Height);
      g.DrawRectangle(new Pen(Color.LightGray, 0), x, y, 1, 1);
  }
  //输出不同字体和颜色的验证码字符
  for (int i=0; i<checkCode.Length; i++)
  {
      int cindex=rand.Next(7);
      int findex=rand.Next(5);
      Font f=new System.Drawing.Font(font[findex],
              10, System.Drawing.FontStyle.Bold);
      Brush b=new System.Drawing.SolidBrush(c[cindex]);
      int ii=4;
      if ((i+1) %2==0)
      {
          ii=2;
      }
      g.DrawString(checkCode.Substring(i, 1), f, b, 3+(i*12), ii);
  }
  //画一个边框
  g.DrawRectangle(new Pen(Color.Black, 0),0,0,image.Width-1,image.Height-1);
  //输出到浏览器
  System.IO.MemoryStream ms=new System.IO.MemoryStream();
  image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
```

```
    HttpContext.Current.Response.ClearContent();
    //Response.ClearContent();
    HttpContext.Current.Response.ContentType="image/Jpeg";
    HttpContext.Current.Response.BinaryWrite(ms.ToArray());
    g.Dispose();
    image.Dispose();
}
```

⑧ 编写一个静态方法供直接调用。代码如下。

```
public static void DrawImage()
{
    CheckCode img=new CheckCode();
    HttpContext.Current.Session["CheckCode"]=img.RndNum(4);
    img.checkCodes(HttpContext.Current.Session["CheckCode"].ToString());
}
```

⑨ 新建空白的 Web 页面(这里不再重复描述如何添加新项,读者应该已经非常熟练掌握了相应操作),按 F7 键切换到代码文件,在 Page_Load 事件中输入 CheckCode.DrawImage()。

⑩ 拖动 Image 控件到验证码文本框右边,单击 ImageUrl 属性,单击右边的选择按钮,打开“选择图像”对话框,在“文件类型”下拉列表中选择“所有文件”选项,单击 CheckCode.aspx页面,单击“确定”按钮。

⑪ 拖动 CustomValidator 控件到“账号”文本框右边,单击 ControlToValidator 属性,单击下拉列表中 TxtAccount 选项,单击 ErrorMessage 属性,输入“用户已经存在”。双击 CustomValidator1控件,进入 CvAccount_ServerValidate 事件代码,代码中需要通过 ADO.NET 访问 T_Customer 表。如果用户存在,则设置 args.IsValid = false。

```
String sql=String.Format("select customer_id from T_Customer
        where customer_account='{0}'",args.Value);
String connectionString="Data Source=.;Initial Catalog=Smart;
        Integrated Security=True;MultipleActiveResultSets=true";
SqlConnection con=new SqlConnection(connectionString);
con.Open();
SqlCommand cmd=new SqlCommand(sql,con);
SqlDataReader sdr=sh.QueryOperation(sql);
if (sdr.HasRows)
{
    args.IsValid=false;
}
else
{
    args.IsValid=true;
}
```

3.4.3 任务 3:会员注册信息存储

1. 任务目标

能使用 ADO.NET 访问数据库。

2. 任务内容

能使用 ADO. NET 技术将会员注册信息保存到 T_Customer 表中。

3. 任务实施步骤

任务 2 中已经对会员输入的信息进行了验证，本任务则需要将页面中的输入信息保存到 T_Customer 表中。

① 拖动 Label 控件到“新用户注册”按钮下方，单击 Text 属性，输入“添加新用户错误”提示信息。

② 双击“新用户注册”按钮，进入按钮的 Click 事件。“新用户注册”事件首先要判断页面是否通过所有的验证控件验证，接着要判断验证码是否输入正确；然后使用 ADO. NET 技术执行非查询操作。示例代码如下。

```
if (Page.IsValid)
{
    if (TxtCheckCode.Text.Equals(Session["CheckCode"].ToString()))
    {
        String connectionString= "Data Source= .;Initial Catalog= Smart
            Integrated Security= True; MultipleActiveResultSets= true";
        SqlConnection con= new SqlConnection(connectionString);
        con.Open();
        String sql= String.Format("insert into T_Customer values('{0}','{1}',
            {2},'{3}','{4}')",TxtAccount.Text,TxtPwd.Text,TxtAge.Text,TxtPhone.Text,TxtEmail.Text);
        SqlCommand cmd= new SqlCommand(sql,con);
        If(cmd.ExeNonQuery()> 0)
        {
            Response.Redirect("~/Shop/index.html");
        }
        else
        {
            lblTS.Visible= true;
        }
    }
}
```

任务 3 正确执行效果图如图 3-17 所示。

	Customer_ID	Customer_Acco...	Customer_Pwd	Customer_Age	Customer_Phone	Customer_Email
▸	1	LLJ	l	19	88886666	llj@gmail.com
	2	michael	1234	20	86571234	l@gmai.com
	3	michael_hello	12345	20	88667733	2@gmai.com
	4	michael2	12	20	12312321	122@gmail.com
	5	michael_hello3	12	20	88866677	mic@nbcc.cn
	6	michael_hello9	123456	21	95845983	q@q.com

图 3-17 任务 3 正确执行效果图

3.5 总结归纳

第三个子项目主要讲解了 ASP. NET 常用服务器控件、验证控件和 ADO. NET 数据库访问技术。ASP. NET 常用服务器控件较多,这一项目没有一一列举,只是将用到的控件详细讲解其常规用法。Visual Studio 2012 提供了各种验证控件。RequiredFieldValidator 确保用户没有跳过输入控件。RequiredFieldValidator 控件可以与文本框绑定在一起,以便强制用户对文本框进行输入。RangeValidator(范围验证)确保输入的数字在指定的范围内。CompareValidator(比较验证)比较用户输入和其他数值。它可以与一个指定的时间比较,或者与另外的一个控件的属性值比较,同样可以与数据库中的值进行比较。RegularExpressionValidator (正则表达式验证)是最强大的验证控件之一,它可以把用户输入和提供的表达式进行比较。可以使用这个验证控件来检查有效的社会保险号码、电话号码、密码等。CustomValidator(用户自定义验证)如果没有符合需要的控件,那么可以使用 CustomValidator 控件。它能够检查出用户的输入是否违背了由自定义方法所提供的算法。此外介绍了 ADO. NET 的对象模型以及如何使用这些模型开发数据绑定 Web 窗体,内容包括创建连接、数据命令和数据阅读器。

3.6 课后习题



选择题

1. 指定 Label 控件的边框风格,需要设置其(　　)属性。
   A. BorderColor　　B. BackColor　　C. BorderStyle　　D. BorderWidth
2. 要将数据源绑定到控件,需要调用控件的(　　)方法。
   A. Load　　B. DataBind　　C. Dispose　　D. GetType
3. 要掩盖 TextBox 控件中的文本,需要将控件的 TextMode 属性设置为(　　)。
   A. Password　　B. MultiLine　　C. SingleLine　　D. Null
4. 要使文本框最多输入 6 个字符,需要将该控件的(　　)属性值设置为 6。
   A. MaxLength　　B. Columns　　C. Rows　　D. TabIndex
5. 要使 Button 控件不可用,需要将控件的(　　)属性设置为 false。
   A. Enabled　　B. EnableViewState
   C. Visible　　D. CausesValidation
6. DropDownList 被选中选项的索引号被置于(　　)属性中。
   A. SelectedIndex　　B. SelectedItem　　C. SelectedValue　　D. TabIndex
7. DropDownList 控件 Items 集合的 Count 属性值是(　　)。
   A. 选择项的序号　　B. 项的总数目　　C. 选择项的数目　　D. 选择项的值
8. DropDownList1. Items[0]. Text 值是控件的(　　)。
   A. 文本　　B. 选择的文本　　C. 添加的文本　　D. 首项的文本

9. 语句"DropDownList1.Items[0].Selected＝true;"的作用是(　　)。
　A. 使首项被选中　　B. 测试首项是否被选中
　C. 去掉首项的选中状态　　D. 使首项可用
10. RequiredFieldValidator 控件的 ErrorMessage 属性用来(　　)。
　A. 设置错误信息　　B. 设置到验证的控件
　C. 定位错误类型　　D. 启动错误处理程序
11. RequiredFieldValidator 控件的 ControlToValidate 属性用来(　　)。
　A. 设置是否需要验证　　B. 设置到验证的控件
　C. 设置验证方式　　D. 设置验证的数据类型
12. RangeValidator 控件用于验证数据的(　　)。
　A. 类型　　B. 格式　　C. 范围　　D. 正则表达式
13. 要验证文本框中输入的数据是否为合法的邮编,需要使用(　　)验证控件。
　A. RequiredFieldValidator　　B. RangeValidator
　C. CompareValidator　　D. RegularExpressionValidator
14. 要使 RadioButton 控件被选中,需要将其(　　)属性设置为 true。
　A. Enabled　　B. Visible　　C. Checked　　D. AutoPostBack

3.7　同步操练

为便捷用户在格林酒店进行网上预订房间,项目经理要求酒店管理系统里面须具备用户注册功能,对所有字段都必须非空,对身份证号码须进行长度验证。最终效果图如图 3-18 所示。



图 3-18　格林酒店注册功能

项目4 会 员 登 录

4.1 项 目 引 入

项目3已经实现了会员注册功能。自然而然也有会员登录功能相对应，也是 Smart On Line 电子商城中必不可少的一项功能。李明是 Smart On Line 电子商城的一位开发人员，接到的任务是协助开发用户登录功能模块。

4.2 项 目 分 析

经过小组讨论和仔细分析，认为 Smart On Line 电子商城用户登录模块使用ASP. NET 服务器控件来设计界面，采用验证控件对界面输入数据进行非空验证。当用户输入准确的用户名和密码后，则会成功登录。在成功登录过程中需要在会话中保存用户名，以便后续在购买商品的时候验证用户是否已经成功登录。如果成功登录，则跳转到首页，否则在登录页面显示登录错误信息。为提供用户的体验，采用 AJAX 技术进行局部刷新。因此本项目可以分为两部分：第一部分是设计登录界面并进行用户信息的验证；第二部分是采用 AJAX 技术进行局部刷新并加强用户的体验。

4.3 知 识 准 备

开发 Smart On Line 电子商城时，Visual Studio 2012 提供的是一个开发环境，而 ASP. NET是网页应用程序开发的一种工具，其中 ASP. NET 提供了常用的内置对象：简单地说，就是不用实例化而直接可以使用的对象(实际上本身内置对象就可以理解为类库中类已经实例化好的对象)，实现用户和网页间信息的传递。简单地可以理解为：ASP. NET 是用户和网页之间信息交流的一个桥梁，不同的内置对象传递不同的信息而已。ASP. NET 提供的内置对象有 Request、Response、Application、Session、Server 和 Cookies。这些对象使用户更容易收集通过浏览器请求发送的信息、响应浏览器以及存储用户信息，以实现其他特定的状态管理和页面信息的传递。现在常用到的对象主要是：Request、Response、Server、Session 和 Cookie 等。

- Response 对象：通过该对象的属性和方法可以控制如何将服务器端的数据发送到客户端浏览器。

- Request 对象：当客户发出请求并执行 ASP.NET 程序时，客户端的请求信息会包装在 Request 对象中，这些请求信息包括请求报头(Header)、客户端的机器信息，客户端浏览器信息，请求方法(如 POST、GET)、提交的窗体信息等。
- Server 对象：反映了 Web 服务器的各种信息，它提供了服务器可以提供的各种服务。
- Session 对象：负责存储、读取和改变一个特定用户的会话信息。对于每个用户的每次访问，Session 对象都是唯一的。
- Cookie 对象：在 Web 程序设计中，它表示一个长度不超过 4KB 的一个普通的文本文件。这个文件在用户的硬盘上，可以由 Web 浏览器进行访问。

4.3.1　Response 对象

Response 对象用来访问所创建的并能被客户端进行响应对象，同时输出信息到客户端，它提供了标识服务器和性能的 HTTP 变量，发送给浏览器的信息和在 Cookie 中存储的信息。它也提供了一系列用于创建输出页面的方法，如无所不在的 Response.Write 方法。Response 对象的常用属性如下所示。

- BufferOutput：获取或设置一个值，该值指示是否缓冲输出，并在完成处理整个页面之后将其发送。
- Cache：获取 Web 页面的缓存策略。
- Charset：获取或设置输出流的 HTTP 字符集类型。
- IsClientConnected：获取一个值，通过该值指示客户端是否仍连接在服务器上。
- ContentEncoding：获取或设置输出流的 HTTP 字符集。
- TrySkipIisCustomErrors：获取或设置一个值，指定是否支持 IIS 7.0 自定义错误输出。

Response 方法可以输出 HTML 流到客户端，其中包括发送信息到客户端和客户端 URL 重定向，不仅如此，Response 还可以设置 Cookie 的值以保存客户端信息。Response 的常用方法如下。

- Write：向客户端发送指定的 HTTP 流。
- End：停止页面的执行并输出相应的结果。
- Clear：清除页面缓冲区中的数据。
- Flush：将页面缓冲区中的数据立即显示。
- Redirect：客户端浏览器的 URL 地址重定向。

在 Response 的常用方法中，Write 方法是最常用的方法，Write 能够向客户端发送指定的 HTTP 流，并呈现给客户端浏览器。当希望在 Response 对象运行时，能够中途进行停止时，则可以使用 End 方法对页面的执行过程进行停止，示例代码如下所示。

```
for (int i=0; i<100; i++)
{
    if (i<10)
    {
```

```
            Response.Write("当前输出了第'+i +'行<hr/>");
        }
        else
        {
            Response.End();
        }
    }
```

上述代码循环输出 HTML 流"当前输出了第 X 行",当输出到 10 行时,则停止输出,如图 4-1 所示。



图 4-1　Response. End 方法

Redirect 方法通常用于页面跳转,示例代码如下所示。执行下述代码,将会跳转到相应的 URL。

```
Response.Redirect("http://www.sina.com");
```

4.3.2　Request 对象

Request 对象是 HttpRequest 类的一个实例,Request 对象用于读取客户端在 Web 请求期间发送的 HTTP 值。Request 对象常用的属性如下。

- QueryString:获取 HTTP 查询字符串变量的集合。
- Path:获取当前请求的虚拟路径。
- UserHostAddress:获取远程客户端 IP 主机的地址。
- Browser:获取有关正在请求的客户端的浏览器功能的信息。

1. QueryString：请求参数

QueryString 属性是用来获取 HTTP 查询字符串变量的集合，通过 QueryString 属性能够获取页面传递的参数。在超链接中，往往需要从一个页面跳转到另外一个页面，跳转的页面需要获取 HTTP 值来进行相应的操作，例如新闻页面的 news. aspx? id=1。为了获取传递过来的 id 的值，则可以使用 Request 的 QueryString 属性，示例代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (! String.IsNullOrEmpty(Request.QueryString["id"]))
    //如果传递的 id值不为空
    {
        Label1.Text=Request.QueryString["id"];
        //将传递的值赋予标签中
    }
    else
    {
        Label1.Text="没有传递的值";
        //提示没有传递的值
    }
    if (! String.IsNullOrEmpty(Request.QueryString["type"]))
    //如果传递的 type值不为空
    {
        Label2.Text=Request.QueryString["type"];
        //获取传递的 type值
    }
    else
    {
        Label2.Text="没有传递的值";
        //无值时进行相应的编码
    }
}
```

上述代码使用 Request 的 QueryString 属性来接受传递的 HTTP 值，当访问页面路径为"http://localhost:29867/Default. aspx"时，默认传递的参数为空，因为其路径中没有对参数的访问。而"http://localhost:29867/Default. aspx? id=1&type=QueryString&action=get"访问路径显示该地址传递了三个参数，分别为 id=1、type=QueryString 以及 action=get。

2. Path：获取路径

通过使用 Path 的方法可以获取当前请求的虚拟路径，示例代码如下所示。

```
Label3.Text=Request.Path.ToString();
```

当在应用程序开发中使用 Request. Path. ToString()时，就能够获取当前正在被请求的文件的虚拟路径的值。当需要对相应的文件进行操作时，可以使用 Request. Path 的信息进行判断。

3. UserHostAddress：获取 IP 记录

通过使用 UserHostAddress 的方法，可以获取远程客户端 IP 主机的地址，示例代码如下所示。

```
Label4.Text=Request.UserHostAddress;
```

在客户端主机 IP 统计和判断中，可以使用 Request. UserHostAddress 进行 IP 统计和判断。在有些系统中，需要对来访的 IP 进行筛选，使用 Request. UserHostAddress 就能够轻松地判断用户 IP 并进行筛选操作。

4. Browser：获取浏览器信息

通过使用 Browser 的方法，可以判断正在浏览网站的客户端浏览器的版本，以及浏览器的一些信息，示例代码如下所示。

```
Label5.Text=Request.Browser.Type.ToString();
```

这些属性能够获取服务器和客户端的相应信息，也可以通过“?”号进行 HTTP 值的传递和获取，上述代码运行结果如图 4-2 所示。



图 4-2　Request 对象

Request 不仅包括这些常用的属性，还包括其他属性，例如用于获取当前目录在服务器虚拟主机中的绝对路径(如 ApplicationPath)。另外，开发人员也可以使用 Request 中的 Form 属性进行页面中窗体的值集合的获取。

示例 1：建立如图 4-3 所示的界面，在列表框中显示获得客户端的信息。

① 如图 4-3 所示，使用一个层和一个 1 行 2 列的表格进行布局，布局结束后拖动一个 RadioButtonList 控件置于布局表格的左侧，通过 RadioButtonList 的集合编辑器为 RadioButtonList 添加 5 个选项，并将 RadioButtonList 控件的 AutoPostBack 属性设为 True。拖动一个 ListBox 控件和一个 Label 控件到布局表格的右侧，将 Label 控件的 Text 属性设置为“来访者相关信息”。

② 本例中仅有 RadioButtonList 控件需要编写代码。双击 RadioButtonList 控件，切换到代码文件，即进入代码编写环境，输入下列代码。

```
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
{
int i;
```

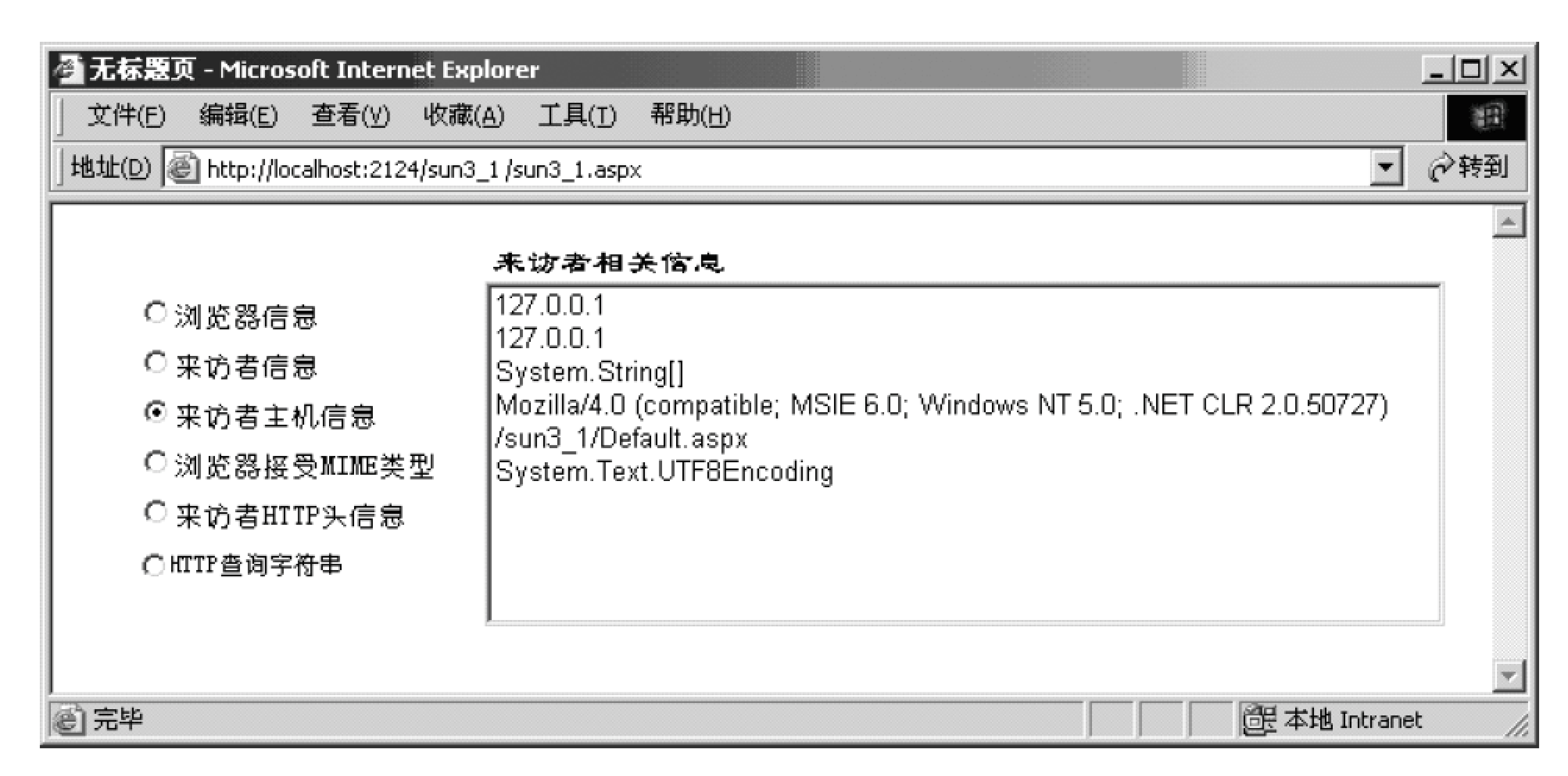


图 4-3　获取来访者的浏览器信息示例

```
switch (RadioButtonList1.SelectedIndex)
{
    case 0:
    {
        ListBox1.Items.Clear();
        ListBox1.Items.Add(Request.Browser.Beta.ToString());
        ListBox1.Items.Add(Request.Browser.Version.ToString());
        ListBox1.Items.Add(Request.Browser.Platform.ToString());
        ListBox1.Items.Add(Request.Browser.Cookies.ToString());
        ListBox1.Items.Add(Request.Browser.ActiveXControls.ToString());
        ListBox1.Items.Add(Request.Browser.Type.ToString());
        ListBox1.Items.Add(Request.Browser.ClrVersion.ToString());
        break;
    }
    case 1:
    {
        ListBox1.Items.Clear();
        ListBox1.Items.Add(Request.UrlReferrer.Port.ToString());
        ListBox1.Items.Add(Request.UrlReferrer.Authority.ToString());
        ListBox1.Items.Add(Request.UrlReferrer.AbsolutePath.ToString());
        ListBox1.Items.Add(Request.UrlReferrer.Host.ToString());
        ListBox1.Items.Add(Request.UrlReferrer.HostNameType.ToString());
        ListBox1.Items.Add(Request.UrlReferrer.UserInfo.ToString());
        break;
    }
    case 2:
    {
        ListBox1.Items.Clear();
        ListBox1.Items.Add(Request.UserHostAddress.ToString());
        ListBox1.Items.Add(Request.UserHostName.ToString());
        ListBox1.Items.Add(Request.UserLanguages[0].ToString());
        ListBox1.Items.Add(Request.UserAgent.ToString());
        ListBox1.Items.Add(Request.Path.ToString());
        ListBox1.Items.Add(Request.ContentEncoding.ToString());
```

```
            break;
        }
        case 3:
        {
            i=Request.AcceptTypes.GetUpperBound(0);
            ListBox1.Items.Clear();
            while(i>0)
            {
                    ListBox1.Items.Add(Request.AcceptTypes[i]);
                    i=i-1;
            }
            break;
        }
        case 4:
        {
            i=Request.Headers.Count-1;
            ListBox1.Items.Clear();
            while (i>0)
            {
                    ListBox1.Items.Add(Request.Headers[i]);
                    i=i - 1;
            }
            break;
        }
        case 5:
        {
            ListBox1.Items.Clear();
            ListBox1.Items.Add(Request.QueryString["ID"]);
            ListBox1.Items.Add(Request.QueryString["Name"]);
            Break
        }
    }
```

本例 switch 语句中有 5 个 case 语句，分别用来在列表框中显示浏览器信息、来访者信息、来访者主机信息、浏览器接收的 MIME 类型和来访者 HTTP 头信息。在 switch 语句块的每个 case 语句中必须包含一个 break 语句。

4.3.3 Session 对象

Session 对象是 HttpSessionState 的一个实例，Session 是用来存储跨页程序的变量或对象，功能基本同 Application 对象一样。但是 Session 对象的特性与 Application 对象不同。Session 对象变量只针对单一网页的使用者，这也就是说各个机器之间的 Session 的对象不尽相同。

例如用户 A 和用户 B，当用户 A 访问该 Web 应用时，应用程序可以显式地为该用户增加一个 Session 值，同时用户 B 访问该 Web 应用时，应用程序同样可以为用户 B 增加一个 Session 值。但是与 Application 不同的是，用户 A 无法存取用户 B 的 Session 值，用户 B 也无法存取用户 A 的 Session 值。Application 对象终止于 IIS 服务停止时，但是 Session 对象

变量终止于联机机器离线时，也就是说当网页使用者关闭浏览器或者网页使用者在页面进行的操作时间超过系统规定时，Session 对象将会自动注销。

1. Session 对象的特性

Session 对象常用的属性如下。

- IsNewSession：如果用户访问页面时创建了新会话，则此属性将返回 true，否则将返回 false。
- TimeOut：传回或设置 Session 对象变量的有效时间，如果在有效时间内没有任何客户端动作，则会自动注销。

Session 对象常用的方法如下。

- Add：创建一个 Session 对象。
- Abandon：该方法用来结束当前会话并清除对话中的所有信息，如果用户重新访问页面，则可以创建新会话。
- Clear：此方法将清除全部的 Session 对象变量，但不结束会话。

Session 对象可以不需要 Add 方法进行创建，直接使用"Session["变量名"]=变量值"的语法也可以进行 Session 对象的创建。

2. Session 对象的使用

Session 对象可以用于安全性较高的场合，例如后台登录。在后台登录的制作过程中，管理员拥有一定的操作时间，而如果管理员在这段时间不进行任何操作，为了保证安全性，后台将自动注销，如果管理员需要再次进行操作，则需要再次登录。在管理员登录时，如果登录成功，则需要给管理员一个 Session 对象，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    Session["admin"]="michael";
    Response.Redirect("default.aspx");
}
```

当管理员单击"注销"按钮时，则会注销 Session 对象并提示再次登录，示例代码如下所示。

```
protected void Button2_Click(object sender, EventArgs e)
{
    Session.Clear();
}
```

在 Page_Load 方法中，可以判断是否已经存在 Session 对象。如果存在 Session 对象，则说明管理员当前的权限是正常的；而如果不存在 Session 对象，则说明当前管理员的权限可能是错误的，或者是非法用户正在访问该页面，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] ! =null)
    {
        if (String.IsNullOrEmpty(Session["admin"].ToString()))
        {
            Button1.Visible=true;
```

```
                Button2.Visible= false;
                //Response.Redirect("admin_login.aspx");
            }
            else
            {
                Button1.Visible= false;
                Button2.Visible= true;
            }
        }
    }
```

上述代码当管理员没有登录时，则会出现“登录”按钮。如果登录了并存在 Session 对象，则“登录”按钮被隐藏，只显示“注销”按钮。

4.3.4 Server 对象

Server 对象是 HttpServerUtility 的一个实例，该对象可以对服务器上的方法和属性进行访问。

1. Server 对象的常用属性

Server 对象的常用属性如下所示。

- MachineName：获取远程服务器的名称。
- ScriptTimeout：获取和设置“请求超时”的信息。

2. Server 对象的常用方法

Server 对象的常用方法如下所示。

- CreateObject：创建 COM 对象的一个服务器实例。
- Execute：使用另一个页面执行当前请求。
- Transfer：终止当前页面的执行，并为当前请求开始执行新页面。
- HtmlDecode：对已编码且消除了 HTML 无效字符的字符串进行解码。
- HtmlEncode：对要在浏览器中显示的字符串进行编码。
- MapPath：返回与 Web 服务器上的执行虚拟路径相对应的物理文件路径。
- UrlDecode：对字符串进行解码，该字符串是为了进行 HTTP 传输而进行了编码，并在 URL 中发送到服务器。
- UrlEncode：编码字符串，以便通过 URL 从 Web 服务器到客户端浏览器进行字符串的传输。

在创建文件、删除文件或者读取文件类型的数据库时（如 Access 和 SQLite），都需要指定文件的路径并显式地提供物理路径执行文件的操作，如 D:\Program Files。但是这样做却暴露了物理路径。如果有非法用户进行非法操作，很容易就显示了物理路径，这样就造成了安全问题。而如果在使用文件和创建文件时，如果一定要为创建文件的保存地址设置一个物理路径，这样非常不方便，并且用户体验也不好。当用户需要上传文件时，用户不可能知道也不应该知道服务器路径。使用 MapPath 方法能够获取相应路径。MapPath 方法以“/”开头，则返回 Web 应用程序的根目录所在的路径；若 MapPath 方法以“../”开头，则会从当前目录开始寻找上级目录，如图 4-4 所示，而其实际服务器路径如图 4-5 所示。





图 4-4　MapPath 示意图

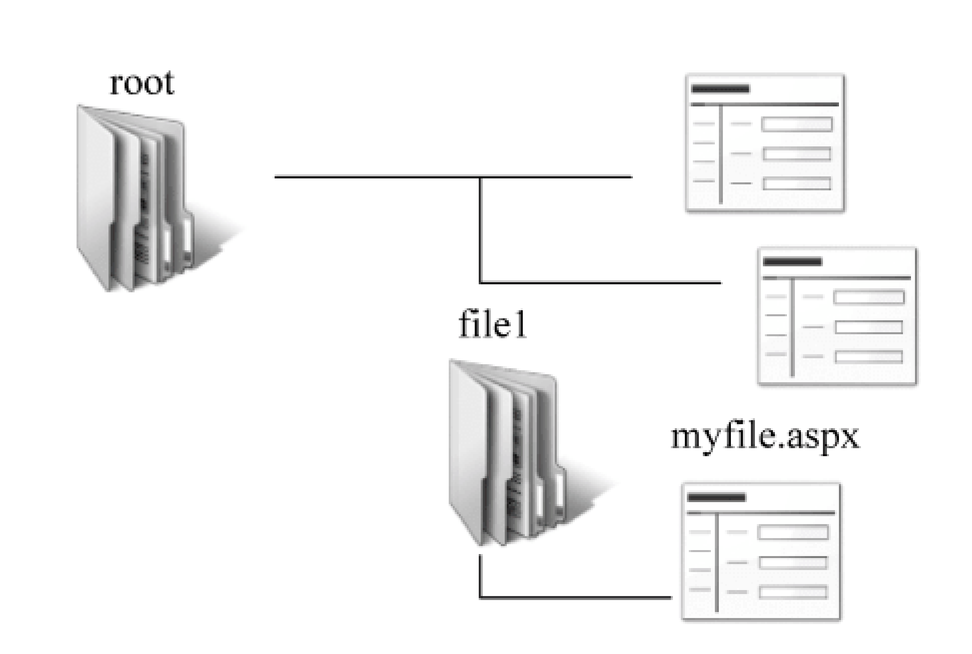


图 4-5　服务器路径

在图 4-4 所示,其中论坛根目录为 root,在根目录下有一个文件夹为 file1,在 file1 中的文件可以使用 MapPath 访问根目录中文件的方法有 Server. MapPath("../文件名称")或 Server. MapPath("/文件名称"),示例代码如下所示。

```
string FilePath=Server.MapPath("../Default.aspx");
//设置路径
string FileRootPath=Server.MapPath("/Default.aspx");
//设置路径
```

Server. MapPath 其实返回的是物理路径,但是通过 MapPath 进行封装后,再通过代码就无法看见真实的物理路径了,若需要知道真实的物理路径,只需输出 Server. MapPath 即可。

4.3.5　Cookie 对象

Session 对象能够保存用户信息,但是 Session 对象并不能够持久地保存用户信息,当用户在限定时间内没有任何操作时,用户的 Session 对象将被注销和清除,在持久化保存用户信息时,Session 对象并不适用。

1. Cookie 对象

使用 Cookie 对象能够持久化地保存用户信息,相对于 Session 对象和 Application 对象而言,Cookie 对象保存在客户端,而 Session 对象和 Application 对象保存在服务器端,所以 Cookie 对象能够长期保存。Web 应用程序可以通过获取客户端的 Cookie 值并判断用户的身份来进行认证。ASP. NET 内包含两个内部的 Cookie 集合,通过 HttpRequest 的 Cookies 集合可以访问客户端信息。Cookie 不是 Page 类的子类,所以用法与 Session 和 Application 不同。相对于 Session 和 Application 而言,Cookie 的优点如下。

- 可以配置到期结束的规则:Cookie 可以在浏览器会话结束后立即到期,也可以在客户端中无限保存。
- 简单:Cookie 是一种基于文本的轻量级结构,包括简单的键值对。
- 数据持久性:Cookie 能够在客户端上长期进行数据的保存。
- 无需任何服务器资源:Cookie 只存储在本地客户端中。

虽然 Cookie 包括若干优点,这些优点能够弥补 Session 对象和 Application 对象的不

足,但是 Cookie 对象同样有缺点,Cookie 的缺点如下。

- 大小限制:Cookie 有大小限制,同时不能无限保存 Cookie 文件。
- 不确定性:如果客户端禁用 Cookie 配置,则 Web 应用中使用的 Cookie 将被限制,客户端将无法保存 Cookie。
- 安全风险:现在有很多的软件能够伪装成 Cookie,这意味着保存在本地的 Cookie 并不安全,同时会导致 Web 应用在认证用户权限时出现错误。

Cookie 是一个轻量级的内置对象,Cookie 并不能将复杂和庞大的文本进行存储,在进行相应的信息或状态的存储时,应该考虑 Cookie 的大小限制和不确定性。

2. Cookie 对象的属性

Cookie 对象的属性如下。

- Name:获取或设置 Cookie 的名称。
- Value:获取或设置 Cookie 的值。
- Expires:获取或设置 Cookie 过期的日期和事件。
- Version:获取或设置 Cookie 符合 HTTP 维护状态的版本。

3. Cookie 对象的方法

Cookie 对象的方法如下。

- Add:增加 Cookie 变量。
- Clear:清除 Cookie 集合内的变量。
- Get:通过变量名称或索引得到 Cookie 的变量值。
- Remove:通过 Cookie 变量名称或索引删除 Cookie 对象。

4. 创建 Cookie 对象

通过 Add 方法能够创建一个 Cookie 对象,并通过 Expires 属性设置 Cookie 对象在客户端中所持续的时间,示例代码如下。

```
HttpCookie MyCookie=new HttpCookie("MyCookie");
MyCookie.Value=Server.HtmlEncode("Cookie应用程序");
//设置 Cookie的值
MyCookie.Expires=DateTime.Now.AddDays(6);
//设置 Cookie过期的时间
Response.AppendCookie(MyCookie);
//新增 Cookie
```

上述代码创建了一个名称为 MyCookie 的 Cookies,上述代码通过使用 Response 对象的 AppendCookie 方法进行 Cookie 对象的创建。

5. 获取 Cookie 对象

Web 应用在客户端浏览器创建 Cookie 对象之后,可以通过 Cookie 的方法读取客户端中保存的 Cookies 信息,示例代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    try
    {
        HttpCookie MyCookie=new HttpCookie("MyCookie");
```

```
            MyCookie.Value=Server.HtmlEncode("我的 Cookie 应用程序");
            MyCookie.Expires=DateTime.Now.AddDays(5);
            Response.AppendCookie(MyCookie);
            Response.Write("Cookies 创建成功");
            Response.Write("<hr/>获取 Cookie 的值<hr/>");
            HttpCookie GetCookie=Request.Cookies["MyCookie"];
            Response.Write("Cookies 的值:"+GetCookie.Value.ToString()
                +"<br/>");        //输出 Cookie 值
            Response.Write("Cookies 的过期时间:"+
                GetCookie.Expires.ToString()+"<br/>");
        }
        catch
        {
            Response.Write("Cookies 创建失败");
        }
    }
```

上述代码创建了一个 Cookie 对象之后，立即获取刚才创建的 Cookie 对象的值和过期时间。使用 Request. Cookies 方法可以通过 Cookie 对象的名称或者索引获取 Cookie 的值。在一些网站或论坛中经常用到 Cookie，当用户浏览并登录网站后，如果用户浏览完毕并退出网站时，Web 应用可以通过 Cookie 方法对用户信息进行保存。当用户再次登录时，可以直接获取客户端的 Cookie 值而无需用户再次进行登录操作。

4.3.6　ASP. NET AJAX

现今，在 Web 开发领域最流行的就属于 AJAX，AJAX 能够提升用户体验，更加方便地与 Web 应用程序进行交互。在传统的 Web 开发中，对页面进行操作往往需要进行“回发”，从而导致页面刷新，而使用 AJAX 就无需产生“回发”从而实现无刷新效果。在 C/S 应用程序的开发过程中，很容易做到无“刷新”样式控制，因为 C/S 应用程序往往是安装在本地的，所以 C/S 应用程序能够维持客户端状态，对于状态的改变能够及时捕捉。相比之下，Web 应用属于一种无状态的应用程序，在 Web 应用程序操作过程中，需要通过 POST 等方法进行页面参数传递，这样就不可避免地产生页面的刷新。在传统的 Web 开发过程中，浏览者浏览一个 Web 页面并进行相应的页面填写时，就需要使用表单向服务器提交信息。当用户提交表单时，就不可避免地会向服务器发送了一个请求，服务器接受该请求并执行相应的操作后，将生成一个页面返回给浏览者。然而，在服务器处理表单并返回新的页面的同时，浏览者第一次浏览时的页面(这里可以当作是旧的页面)和服务器处理表单后返回的页面在形式上基本相同，当大量的用户进行表单提交操作时，无疑增加了网络的带宽，因为处理前和处理后的页面基本相同。

在 C/S 应用程序开发中，C/S 应用程序往往安装在本地，这样响应用户事件的时间非常短，而且 C/S 应用程序能够及时捕捉相应用户的操作，而在 Web 端，由于每次的交互都需要向服务器发送请求，服务器接受请求和返回请求的过程就依赖于服务器的响应时间，所以给用户的感觉是访问速度比在本地慢得多。为了解决这一问题，可以在用户浏览器和服务器之间设计一个中间层——AJAX 层。AJAX 改变了传统的 Web 中客户端和服务器的“请求—等待—请求—等待”的模式，通过使用 AJAX 应用向服务器发送和接收需要的数据，从而不会产生页面的刷新。AJAX 应用通过使用 SOAP 或其他一些基于 XML 的

Web Service 接口，并在客户端采用 JavaScript 处理来自服务器的响应，减少了服务器和浏览器之间的“请求—回发”操作，从而减少了带宽。当服务器和客户端之间的信息通信减少之后，浏览者就会感觉到 Web 应用中的操作就更快了。AJAX 将一些应用的处理交付给客户端，让服务器端原本应该运行的操作和需要处理的事务分布给客户端，这样服务器端的处理时间也减少了。

相对于传统的 Web 开发，AJAX 提供了更好的用户体验，AJAX 也提供了较好的 Web 应用交互的解决方案。AJAX 的核心是 JavaScript 对象 XmlHttpRequest。该对象在 Internet Explorer 5 中就被引入了，它是一种支持异步请求的技术。简而言之，XmlHttpRequest 使用户可以使用 JavaScript 向服务器提出请求并处理响应，而不会影响客户端的信息通信。传统的 Web 应用和 AJAX 应用模型如图 4-6 所示。

AJAX Web 应用模型的优点在于，无需进行整个页面的回发就能够进行局部的更新，这样能够使 Web 服务器能够尽快地响应用户的要求。而 AJAX Web 应用无需安装任何插件，也无需在 Web 服务器中安装应用程序，但是 AJAX 需要用户允许 JavaScript 在浏览器上执行，如图用户不允许 JavaScript 在浏览器上执行，则 AJAX 可能无法运行。AJAX 技术看似非常复杂，其实 AJAX 并不是新技术，AJAX 只是一些老技术的混合体，AJAX 通过将这些技术进行一定的修改、整合，就形成了 AJAX 技术。这些老技术包括 XHTML、CSS、DOM、JavaScript、XML 等编程技术。上面的这些技术并不是最新的技术，这些技术已经在现在的开发中被普遍使用，包括 XHTML、CSS 和 DOM，开发人员能够使用 JavaScript 进行 Web 应用中 Web 编程和客户端状态维护，而通过使用 XML 技术能够进行数据保存和交换。除了上面的一些老技术，AJAX 还包含另一次技术，这次技术就是 XMLHttpRequest。在 AJAX 中，最重要的就是 XMLHttpRequest 对象，XMLHttpRequest 对象是 JavaScript 对象，正是 XMLHttpRequest 对象使 AJAX 可以在服务器和浏览器之间通过 JavaScript 创建一个中间层，从而实现了异步通信，如图 4-7 所示。

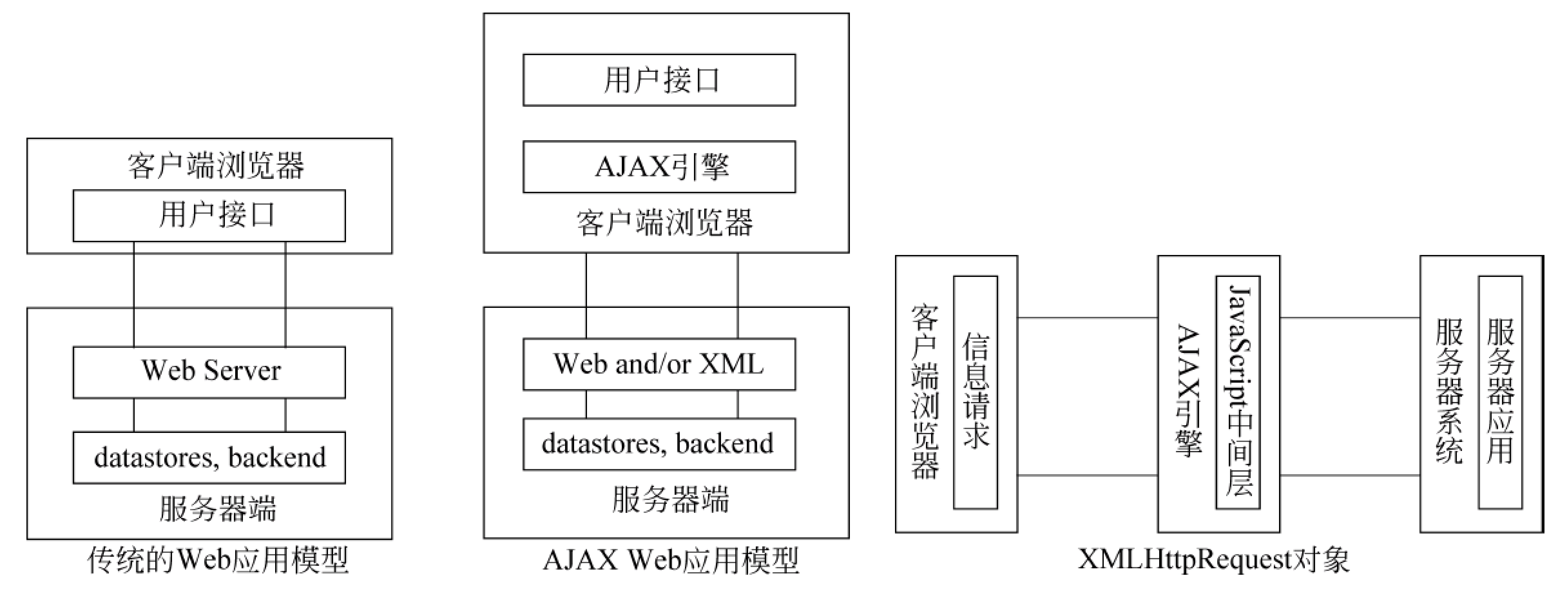


图 4-6　传统 Web 应用和 AJAX Web 应用模型　　图 4-7　XMLHttpRequest 对象的实现过程

AJAX 通过使用 XMLHttpRequest 对象实现异步通信。使用 AJAX 技术后，例如当用户填写一个表单，数据并不是直接从客户端发送到服务器，而是通过客户端发送到一个中间层，这个中间层可以被称为 AJAX 引擎。开发人员无需知道 AJAX 引擎是如何将数据发送到服务器的。当 AJAX 引擎将数据发送到服务器时，服务器同样也不会直接将数据返回给

浏览器，而是通过 JavaScript 中间层将数据返回给客户端浏览器。XMLHttpRequest 对象使用 JavaScript 代码可以自行与服务器进行交互。在 ASP. NET 2. 0 中，AJAX 需要下载和安装，开发人员还需要将相应的 DLL 文件分类存放并配置 Web. config 文件才能够实现 AJAX 功能。而在 ASP. NET4. 0 中，AJAX 已经成为. NET 框架的原生功能，可以直接拖动 AJAX 控件进行 AJAX 开发。AJAX 能够同普通控件一起使用，从而实现 ASP. NET 4. 0 AJAX 中页面无刷新功能。虽然 AJAX 的原理听上去非常复杂，但是 AJAX 的使用却是非常方便。ASP. NET 4. 0 提供了 AJAX 控件以便开发人员快速地进行 AJAX 应用程序开发。在进行 AJAX 页面开发时，首先需要使用脚本管理控件(ScriptManger)，示例代码如下所示。

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

开发人员无需对 ScriptManger 控件进行配置，只需保证 ScriptManger 控件在 UpdatePanel 控件之前即可。使用了 ScriptManger 控件之后，可以使用 UpdatePanel 来确定需要进行局部更新的控件，示例代码如下所示。

```
<form id="form1" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <div>
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:Button ID="Button1" runat="server"
                    Text="得到当前时间" onclick="Button1_Click" />
            </div>
        </ContentTemplate>
    </asp:UpdatePanel>
</form>
```

上述代码使用了 UpdatePanel 控件将服务器控件进行绑定，当浏览者操作 UpdatePanel 控件中的控件实现某种特定的功能时，页面只会针对 UpdatePanel 控件之间的控件进行刷新操作，而不会进行这个页面的刷新。使用 UpdatePanel 控件后，当单击“得到当前时间”按钮，页面只会针对 UpdatePanel 控件内的内容进行更新，而不会影响 UpdatePanel 控件外的控件，运行后如图 4-8 所示。



图 4-8　得到当前时间

在 ASP.NET 4.0 中，系统提供了 AJAX 控件以便开发人员能够在 ASP.NET 4.0 中进行 AJAX 应用程序开发，通过使用 AJAX 控件能够减少大量的代码开发，为开发人员提供了 AJAX 应用程序搭建和应用的绝佳环境。

1. 脚本管理控件(ScriptManger)

脚本管理控件(ScriptManger)是 ASP.NET AJAX 中非常重要的控件，通过使用 ScriptManger，能够进行整个页面局部更新的管理。ScriptManger 用来处理页面上局部的更新，同时生成相关的代理脚本以便能够通过 JavaScript 访问 Web Service。ScriptManger 只能在页面中被使用一次，这也就是说每个页面只能使用一个 ScriptManger 控件，ScriptManger 控件用来进行该页面的全局管理。ScriptManger 控件的常用属性如下所示。

- AllowCustomErrorRedirect：指明在异步回发过程中是否进行自定义错误重定向。
- AsyncPostBackTimeout：指定异步回发的超时事件，默认为 90 秒。
- EnablePageMethods：是否启用页面方法，默认值为 Flase。
- EnablePartialRendering：在支持的浏览器上为 UpdatePanel 控件启用异步回发。默认值为 True。
- LoadScriptsBeforeUI：指定在浏览器中呈现 UI 之前是否应加载脚本引用。
- ScriptMode：指定要在多个类型时可加载的脚本类型，默认为 Auto。

在 AJAX 应用中，ScriptManger 控件基本不需要配置就能够使用。因为 ScriptManger 控件通常需要同其他 AJAX 控件搭配使用。在 AJAX 应用程序中，ScriptManger 控件就相当于一个总指挥官，这个总指挥官只是进行指挥，而不进行实际的操作。

2. 管理控件(ScriptMangerProxy)

ScriptManger 控件作为整个页面的管理者，ScriptManger 控件能够提供强大的功能，使开发人员无需关心 ScriptManger 控件是如何实现 AJAX 功能的，但是一个页面只能使用一个 ScriptManger 控件，如果在一个页面中使用多个 ScriptManger 控件则会出现异常。在 Web 应用的开发过程中，常常需要使用到母版页。如同前面内容提到，母版页和内容窗体一同组合成为一个新页面并呈现在客户端浏览器中，那么如果在母版页中使用了 ScriptManger 控件，而在内容窗体中也使用 ScriptManger 控件，整合在一起的页面就会出现错误。为了解决这个问题，就可以使用另一个脚本管理控件，即 ScriptMangerProxy 控件。ScriptMangerProxy 控件与 ScriptManger 控件非常相似，但是 ScriptManger 控件只允许在一个页面中使用一次。当 Web 应用需要使用母版页进行样式控制时，母版页和内容页都需要进行局部更新时 ScriptManger 控件就不能完成需求，使用 ScriptMangerProxy 控件就能够在母版页和内容页中都实现 AJAX 应用。

3. 时间控件(Timer)

在 C/S 应用程序开发中，Timer 控件是最常用的控件，使用 Timer 控件能够进行时间控制。Timer 控件被广泛地应用在 Windows WinForm 应用程序开发中，Timer 控件能够在一定的时间内间隔地触发某个事件，例如每隔 5s 就执行某个事件。但是在 Web 应用中，由于 Web 应用是无状态的，开发人员很难通过编程方法实现 Timer 控件，虽然 Timer 控件还是可以通过 JavaScript 实现，但是也是以复杂的编程为代价的，这样就造成了 Timer 控件的使用困难。在 ASP.NET AJAX 中，AJAX 提供了一个 Timer 控件，用于执行局部更新，使用 Timer 控件能够控制应用程序在一段时间内进行事件刷新。Timer 控件初始代码如下所示。

```
<asp:Timer ID="Timer1" runat="server">
</asp:Timer>
```

开发人员能够通过设置 Timer 控件的属性进行相应事件的触发，Timer 的属性如下所示。

- Enabled：是否启用 Tick 时间引发。
- Interval：设置 Tick 事件之间的连续时间，单位为 ms。

通过设置 Timer 控件的 Interval 属性，能够指定 Time 控件在一定时间内进行事件刷新操作，示例代码如下所示。

```
<form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="Label1"runat="server"Text="Label"></asp:Label>
                <asp:Timer ID="Timer1" runat="server"
                    Interval="5000" ontick="Timer1_Tick">
                </asp:Timer>
            </ContentTemplate>
        </asp:UpdatePanel>
    </div>
    </form>
```

上述代码使用了一个 ScriptManage 控件进行页面全局管理，ScriptManage 控件是必须的。另外，在页面中使用了 UpdatePanel 控件，该控件用于控制页面的局部更新，而不会引发整个页面更新。在 UpdatePanel 控件中，包括一个 Label 控件和一个 Timer 控件，Label 控件用于显示时间，Timer 控件每 5s 执行一次 Timer1_Tick 事件，Timer 控件的事件代码如下所示。

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    Label1.Text=DateTime.Now.ToString();
}
```

上述代码在页面被呈现时，将当前时间传递并呈现到 Label 控件中，Timer 控件每隔 5s 进行一次刷新，这样就形成了一个可以计数的时间，如图 4-9 所示。



图 4-9　刷新操作

Timer 控件能够通过简单的方法让开发人员无需通过复杂的 JavaScript 代码而实现 Timer 控制。但是从另一方面来讲，Timer 控件会占用大量的服务器资源，如果不停地进行客户端服务器的信息通信操作，很容易造成服务器当机。

4. 更新区域控件(UpdatePanel)

更新区域控件(UpdatePanel)在 ASP. NET AJAX 中是最常用的控件，在上面几节中已经用到 UpdatePanel 控件。UpdatePanel 控件的使用方法同 Panel 控件类似，只需要在 UpdatePanel 控件中放入需要刷新的控件就能够实现局部刷新。使用 UpdatePanel 控件时，整个页面中只有 UpdatePanel 控件中的服务器控件或事件会进行刷新操作，而页面的其他地方都不会被刷新。UpdatePanel 控件可以用来创建局部更新，开发人员无需编写任何客户端脚本。UpdatePanel 控件的属性如下所示。

- RenderMode：该属性指明 UpdatePanel 控件内呈现的标记应为<div>或<span>。
- ChildrenAsTriggers：该属性指明来自 UpdatePanel 控件的子控件的回发是否会导致 UpdatePanel 控件的更新，其默认值为 True。
- EnableViewState：指明控件是否自动保存其往返过程。
- Triggers：指明可以导致 UpdatePanel 控件更新的触发器的集合。
- UpdateMode：指明 UpdatePanel 控件回发的属性，是在每次触发事件时进行更新还是使用 UpdatePanel 控件的 Update 方法再进行更新。
- Visible：UpdatePanel 控件的可见性。

UpdatePanel 控件要进行动态更新，必须依赖于 ScriptManage 控件。当 ScriptManage 控件允许局部更新时，它会以异步的方式发送到服务器，服务器接受请求后，执行操作并通过 DOM 对象来替换局部代码，其原理如图 4-10 所示。

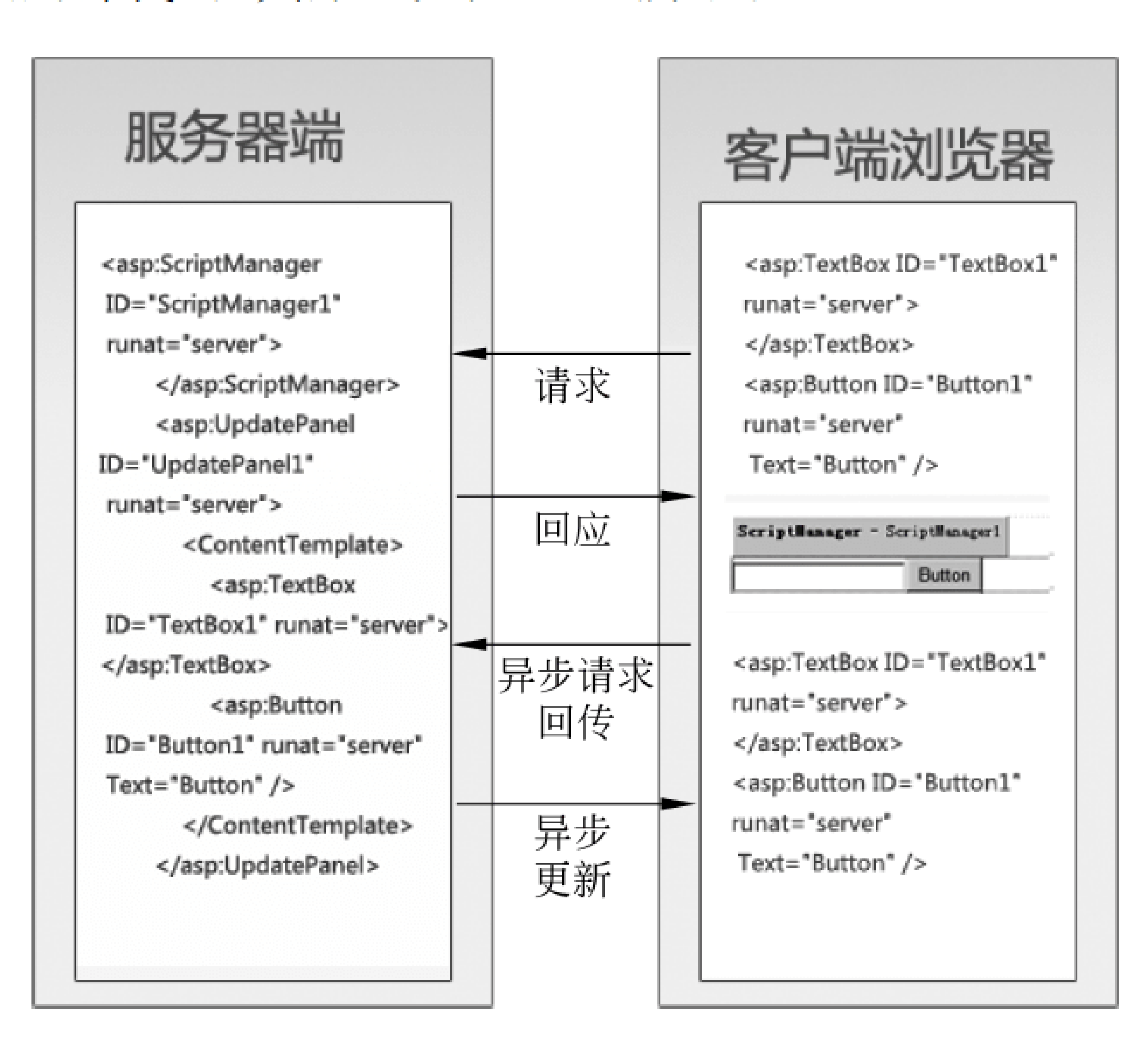


图 4-10　UpdatePanel 控件异步请求示意图

UpdatePanel 控件包括 ContentTemplate 标签。在 UpdatePanel 控件的 ContentTemplate 标签中，开发人员能够放置任何 ASP. NET 控件到 ContentTemplate 标

签中,这些控件就能够实现页面无刷新的更新操作。

5. 更新进度控件(UpdateProgress)

使用 ASP. NET AJAX 常常会给用户造成疑惑。例如当用户进行评论或留言时,页面并没有刷新,而是进行了局部刷新,这个时候用户很可能不清楚到底发生了什么,以至于用户很有可能会产生重复操作,甚至会产生非法操作。更新进度控件(UpdateProgress)就用于解决这个问题,当服务器端与客户端进行异步通信时,需要使用 UpdateProgress 控件告诉用户现在正在执行中。例如当用户进行评论时,当用户单击按钮提交表单,系统应该提示“正在提交中,请稍候”,这样就提供了便利从而让用户知道应用程序正在运行中。这种方法不仅能够让用户操作时更少地出现错误,也能够提升用户体验的友好度。UpdateProgress 控件的 HTML 代码如下所示。

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
        <ProgressTemplate>
            正在提交中,请稍候...<br />
        </ProgressTemplate>
</asp:UpdateProgress>
```

上述代码定义了一个 UpdateProgress 控件,并通过使用 ProgressTemplate 标记进行等待中的样式控制。ProgressTemplate 标记用于标记等待中的样式。当用户单击按钮进行相应的操作后,如果服务器和客户端之间需要时间等待,则 ProgressTemplate 标记就会呈现在用户面前,以提示用户应用程序正在运行。完整的 UpdateProgress 控件和 UpdatePanel 控件代码如下所示。

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
        <asp:UpdateProgress ID="UpdateProgress1" runat="server">
            <ProgressTemplate>
                正在提交中,请稍候...<br />
                </ProgressTemplate>
            </asp:UpdateProgress>
                <asp:Label ID="Label1"runat="server"Text="Label"></asp:Label>
                <asp:Button ID="Button1" runat="server" Text="Button"
                onclick="Button1_Click" />
        </ContentTemplate>
</asp:UpdatePanel>
```

上述代码使用了 UpdateProgress 控件用户进度更新提示,同时创建了一个 Label 控件和一个 Button 控件,当用户单击 Button 控件时则会提示用户正在更新,Button 更新事件代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(3000);
    Label1.Text=DateTime.Now.ToString();
}
```

上述代码使用了 System. Threading. Thread. Sleep 方法指定系统线程挂起的时间，这里设置为 3000ms，这也就是说当用户进行操作后，在这 3000ms 的时间内会呈现“正在提交中，请稍候...”几个字，当 3000ms 过后，就会执行下面的方法。

6. 母版页刷新内容窗体

在母版页中使用 ScriptManage 控件能够方便地将整个页面进行 AJAX 全局控制。当 Web 应用中有很多相似页面，又需要执行 AJAX 应用时，可以在母版页中使用 ScriptManage 控件，在内容窗体中使用 UpdatePanel 控件，这样母版页中的 ScriptManage 控件也会整合到内容窗体中并进行输出。同样，母版页也能够刷新内容窗体中的控件信息。如果 ASP. NET AJAX Web 应用中很多的页面都需要执行相同的操作，而内容窗体同样需要执行这些操作，可以通过在母版页中注册异步传送控件以支持内容窗体中 AJAX 应用的需求，这样只需要在内容窗体中进行控件布局，而无需在内容窗体中再次创建局部更新控件和进行事件的编写。

4.4 项 目 实 施

4.4.1 任务 1：登录界面设计和用户信息验证

1. 任务目标

(1) 能熟练使用验证控件验证输入数据。

(2) 能熟练运用 ADO. NET 查询信息。

2. 任务内容

(1) 从 Layout. master 模板页创建内容页。

(2) 使用服务器控件设计登录页面。

(3) 使用 ADO. NET 技术查询 T_Customer 表信息。

3. 任务实施步骤

该任务首先从 Layout. master 母版页创建内容页，再使用 DIV+CSS 进行页面布局，然后使用常用的服务器控件设计登录界面(如图 4-11 所示)。应使用必填验证控件验证确保

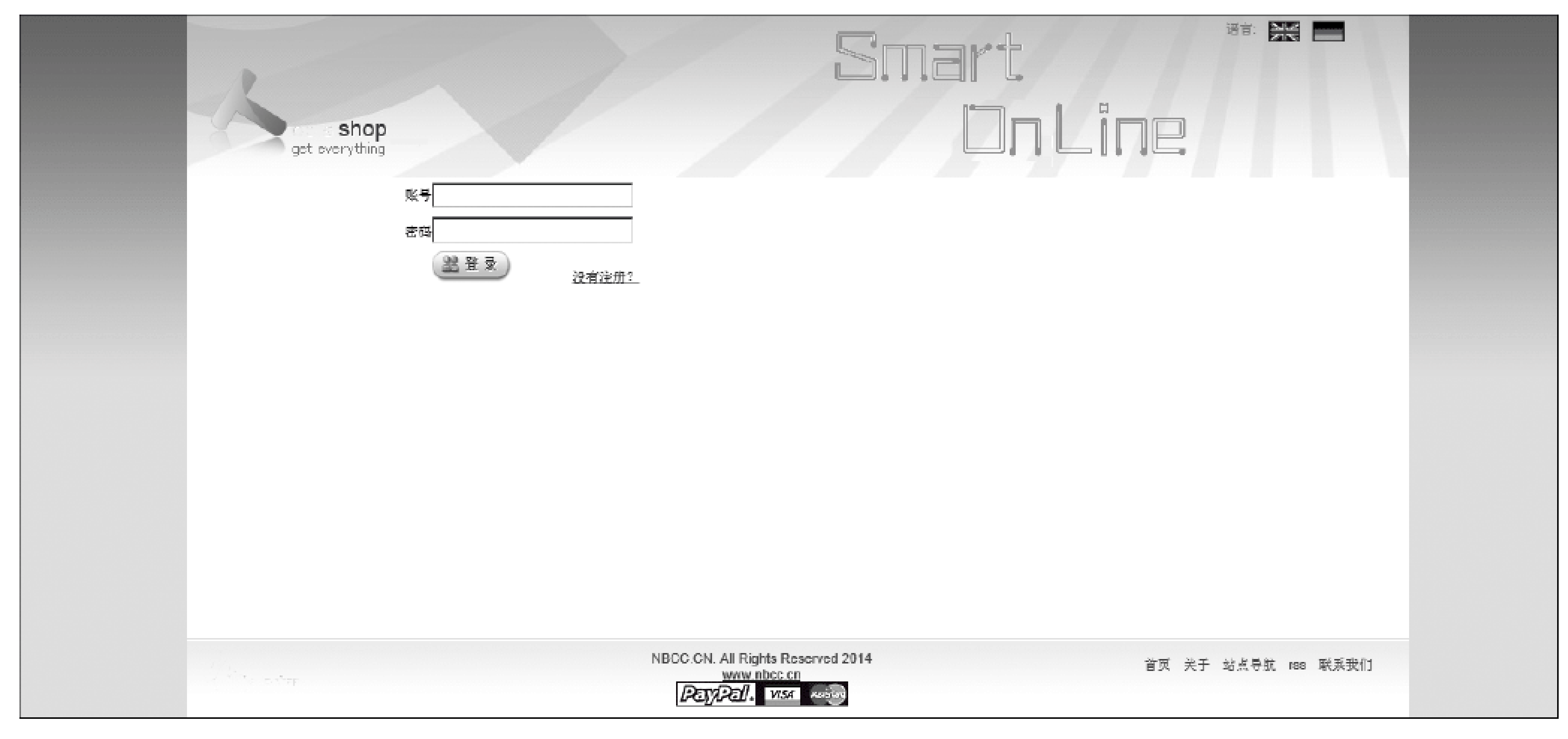


图 4-11　任务 1 效果图

输入信息非空。最后使用 ADO. NET 技术访问 Smart 数据库中的 T_Customer 表。

① 右击 Shop 文件夹,单击"添加"→"添加新项"命令,打开"添加新项"对话框。

② 单击"Web 页面"列表项,在"名称"文本框中输入 Login. aspx,单击"选择母版页",单击"添加"按钮,打开"选择模板页"对话框。单击 Layout. master 文件,再单击"确定"按钮。

③ 使用 DIV 控件完成布局。一个行由两个 DIV 控件控制页面,更改 DIV 的样式。重复该操作,完成三行的布局。

```
<div class="reg_div_tips" >账号</div>
<div class="reg_div_content">
</div>
```

④ 拖动 TextBox 控件到页面相应位置,更改 ID 号。拖动 RequiredFieldValidator 控件到页面中,设置 ControlToValidator 属性和 ErrorMessage 属性,对 TextBox 进行非空验证。

⑤ 拖动 ImageButton 控件到页面相应位置,设置 ImageUrl 属性为"～/gif/07. gif"。

⑥ 拖动 Label 控件到页面相应位置,设置 ForeColor 属性为＃993333。设置 Text 属性为"用户名或者密码错误,请重新输入"。

⑦ 拖动 HyperLink 控件到页面相应位置,设置 Text 属性为"没有输入?",设置 NavigateUrl 属性为"～/Shop/reg. aspx",使其导航到注册页面。完整的页面代码如下。

```
<div class="reg_div">
  <div class="reg_div_tips" >账号</div><div class="reg_div_content">
    <asp:TextBox ID="TxtAccount"
    runat="server" Width="160px"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat= "server"
    ControlToValidate="TxtAccount"
    ErrorMessage="*" "ForeColor="#CC0000"></asp:RequiredFieldValidator>
  </div>
</div>
<div class="reg_div">
  <div class="reg_div_tips" >密码</div><div class="reg_div_content">
      <asp:TextBox ID="TxtPwd" runat="server"
          TextMode="Password" Width="160px"></asp:TextBox>
      <asp:RequiredFieldValidator ID="RequiredFieldValidator2"runat="server"
        ControlToValidate="TxtPwd"
        ErrorMessage="*" ForeColor="#CC0000">
      </asp:RequiredFieldValidator>
  </div>
  </div>
  <div class="reg_div">
    <div class="reg_div_tips" ></div><div class="reg_div_content">
      <asp:ImageButton ID="ImageButton1" runat="server" Height="25px"
        ImageUrl="~/gif/07.gif"
        onclick="ImageButton1_Click" Width="65px" />
          <asp:Label ID="Label1" runat="server"
            ForeColor="#993333"
    Text="用户名或者密码错误,请重新输入" Visible="False">
```

```
        </asp:Label>
          <asp:HyperLink ID="HyperLink1"
            runat="server" NavigateUrl="~/Shop/reg.aspx">没有注册?
          </asp:HyperLink>
            <br />
        </div>
</div>
```

⑧ 登录到 SQL Server 服务器,单击“新建查询”按钮,输入下述代码,按 F5 功能键创建存储过程 QueryCustomer。

```
Create proc QueryCustomer
(
    @c_account nvarchar(50),
    @c_pwd nvarchar(50)
)
as
select customer_id from T_Customer where customer_account=@c_account and customer_pwd=@c_pwd;
```

⑨ 双击“登录”按钮,进入事件代码编辑窗口,输入下列代码。

```
Label1.Visible=false;
SqlConnection con=new SqlConnection(
    System.Configuration.ConfigurationManager.ConnectionStrings
        ["SmartConnectionString"].ToString());
con.Open();
SqlCommand cmd=new SqlCommand("QueryCustomer", con);
cmd.CommandType=System.Data.CommandType.StoredProcedure;
SqlParameter p1=new SqlParameter();
p1.ParameterName="@c_account";
p1.SqlDbType=System.Data.SqlDbType.NChar;
p1.Value=TxtAccount.Text;
cmd.Parameters.Add(p1);
SqlParameter p2=new SqlParameter();
p2.ParameterName="@c_pwd";
p2.SqlDbType=System.Data.SqlDbType.NChar;
p2.Value=TxtPwd.Text;
cmd.Parameters.Add(p2);
SqlDataReader sdr=cmd.ExecuteReader();
if (sdr.HasRows)
{
    //modify session 添加用户账户
    Session.Add("useraccount", TxtAccount.Text);
    Response.Redirect("~/Shop/index.html");
}
else
{
    Label1.Visible=true;
}
```

4.4.2 任务 2：会员输入信息验证

1. 任务目标

能使用 ASP. NET AJAX 进行局部刷新。

2. 任务内容

使用 ASP. NET AJAX 技术让注册页面进行局部刷新。

3. 任务实施步骤

这个任务需要将以前创建的页面代码包含在 UpdatePanel 中。

① 拖动 ScriptManager 控件到 Login. aspx 页面。

② 拖动 UpdatePanel 控件到 Login. aspx 页面。单击“源”按钮，切换到源视图。在<asp:UpdatePanel ID="UpdatePanel1" runat="server">后输入：

```
<ContentTemplate></ContentTemplate>
```

③ 将任务 1 中步骤⑦的页面代码剪切到<ContentTemplate>与</ContentTemplate>之间，如下面的代码所示。

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
        <div class="reg_div">
            <div class="reg_div_tips" >账号</div><div class=
            "reg_div_content">
                <asp:TextBox ID="TxtAccount" runat="server" Width="160px">
                </asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
                runat="server"
                ControlToValidate="TxtAccount" ErrorMessage="* "
                ForeColor="#CC0000"></asp:RequiredFieldValidator>
            </div>
        </div>
        <div class="reg_div">
            <div class="reg_div_tips" >密码</div><div class=
            "reg_div_content">
                <asp:TextBox ID="TxtPwd" runat="server" TextMode="Password"
                Width="160px"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator2"
                runat="server"
                ControlToValidate="TxtPwd" ErrorMessage="* " ForeColor=
                "#CC0000"></asp:RequiredFieldValidator>
            </div>
        </div>
        <div class="reg_div">
            <div class="reg_div_tips" ></div><div class=
            "reg_div_content">
                <asp:ImageButton ID="ImageButton1" runat="server" Height="25px"
                    ImageUrl="~/gif/07.gif"
```

```
                onclick="ImageButton1_Click" Width="65px" />
                <asp:Label ID="Label1" runat="server" ForeColor="#993333"
            Text="用户名或者密码错误,请重新输入" Visible="False"></asp:Label>
        <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl=
        "~/Shop/reg.aspx">没有注册?
        </asp:HyperLink>
        <br />
        </div>
    </div>
    </ContentTemplate>
</asp:UpdatePanel>
```

当用户输入错误的用户名或者密码时,页面对 Login.aspx 页面中的 UpdateContent 包含的部分进行局部刷新,不会对母版页中的内容进行刷新(如图 4-12 所示)。

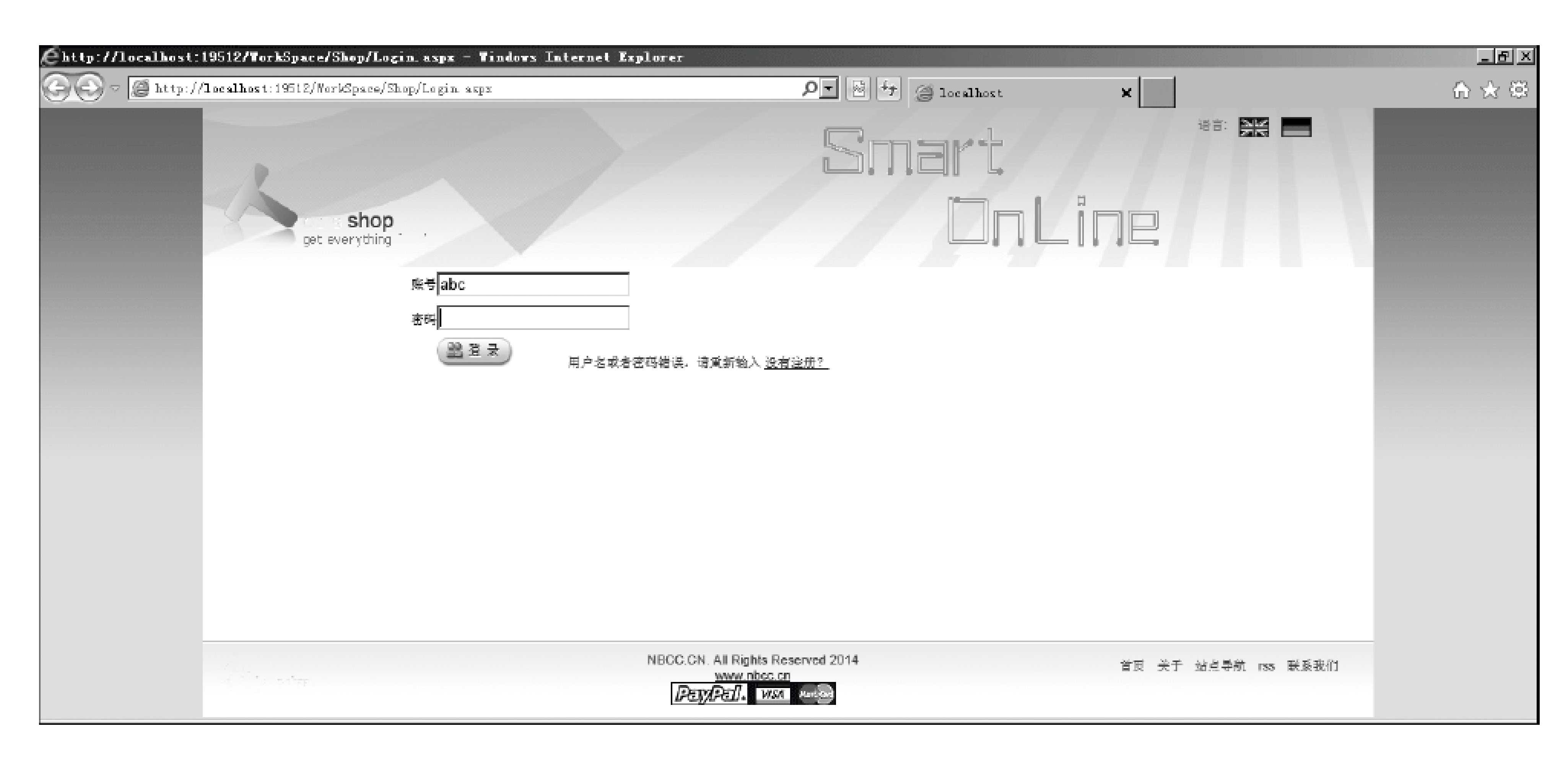


图 4-12 局部刷新

4.5 总 结 归 纳

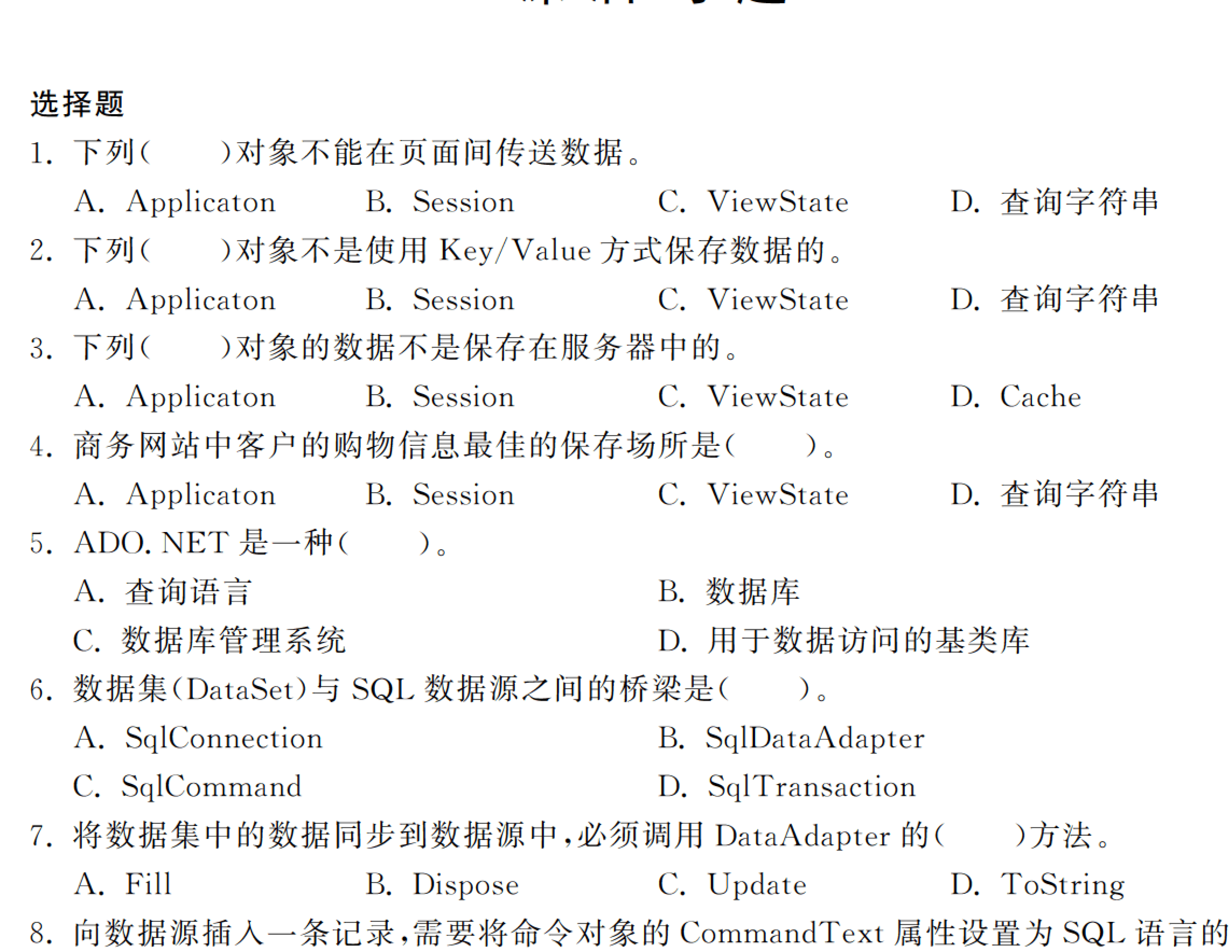
本项目要求开发人员开发登录页面,该任务知识点部分是子项目 3 中详细介绍的,例如验证控件的使用,CSS+DIV 的使用,ADO.NET 访问技术的使用。子项目 4 中主要引入了 ASP.NET 内置对象和 AJAX。Request 对象是用来获取客户端在请求一个页面或传送一个表单(Form)时提供的所有信息,这包括能够标识浏览器和用户的 HTTP 变量,存储在客户端的 Cookie 信息以及附在 URL 后面的值。Response 对象用来访问所创建的客户端的响应,并输出信息到客户端,它提供了标识服务器和性能的 HTTP 变量,发送给浏览器的信息和 Cookie 中存储的信息。它也提供了一系列用于创建输出页面的方法。Cookie 是一小段由浏览器存储在客户端系统上(硬盘)的文本,是一种标记。由 Web 服务器嵌入用户浏览器中,以便标识用户,且随同每次用户的请求发往 Web 服务器。Cookies 的值比 ASP.NET 其他集合的值要复杂得多。

Session 对象就是服务器给客户端的一个编号。当一台 Web 服务器运行时,可能有若干个用户正在浏览这台服务器上的网站。当每个用户首次与这台 WWW 服务器建立连接

时，它就能与这个服务器建立了一个会话（Session），同时服务器会自动为其分配一个SessionID，用于表示这个用户的唯一身份。特别应注意的是，Session 对象的变量只是对一个用户有效，不同的用户的会话信息用不同的 Session 对象的变量存储。在网络环境下，Session 对象的变量是有生命周期的，如果在规定的时间没有对 Session 对象的变量使用，系统就会终止这些变量。

借助 ASP.NET AJAX 控件，使用很少的客户端脚本或不使用客户端脚本就能创建丰富的客户端行为，如在异步回发过程中进行部分页面的更新（在回发时刷新网页的选定部分，而不是刷新整个网页）和显示更新进度。ScriptManager 控件为启用了 AJAX 的 ASP.NET 网页管理客户端脚本。ScriptManagerProxy 控件允许内容页和用户控件等嵌套组件在父元素中已定义了 ScriptManager 控件的情况下将脚本和服务引用添加到网页中。Timer 控件在定义的时间间隔执行回发。如果将 Timer 控件和 UpdatePanel 控件结合在一起使用，可以按照定义的间隔启用部分页更新。还可以使用 Timer 控件来发布整个网页。UpdatePanel 控件可用于生成功能丰富、以客户端为中心的 Web 应用程序。通过使用 UpdatePanel 控件，可以执行部分页面的更新。UpdateProgress 控件提供有关 UpdatePanel 控件中的部分页更新的状态信息。

4.6　课 后 习 题



选择题

1. 下列（　　）对象不能在页面间传送数据。

A. Applicaton　　B. Session　　C. ViewState　　D. 查询字符串

2. 下列（　　）对象不是使用 Key/Value 方式保存数据的。

A. Applicaton　　B. Session　　C. ViewState　　D. 查询字符串

3. 下列（　　）对象的数据不是保存在服务器中的。

A. Applicaton　　B. Session　　C. ViewState　　D. Cache

4. 商务网站中客户的购物信息最佳的保存场所是（　　）。

A. Applicaton　　B. Session　　C. ViewState　　D. 查询字符串

5. ADO.NET 是一种（　　）。

A. 查询语言　　B. 数据库

C. 数据库管理系统　　D. 用于数据访问的基类库

6. 数据集（DataSet）与 SQL 数据源之间的桥梁是（　　）。

A. SqlConnection　　B. SqlDataAdapter

C. SqlCommand　　D. SqlTransaction

7. 将数据集中的数据同步到数据源中，必须调用 DataAdapter 的（　　）方法。

A. Fill　　B. Dispose　　C. Update　　D. ToString

8. 向数据源插入一条记录，需要将命令对象的 CommandText 属性设置为 SQL 语言的 INSERT 命令后，再调用命令对象的（　　）方法。

A. ExecuteNonQuery　　B. ExecuteReader

C. ExecuteScalar　　D. ExecuteXmlReader()

9. 以下(　　)技术不是 AJAX 的常用技术。

A. JavaScript　　B. XML　　C. CSS　　D. Cache

10. 在 ASP.NET 的 AJAX 框架技术中,要实现无刷新,需要采用下面的(　　)控件。

A. UpdatePanel　　B. AutoComplete

C. Timer　　D. Rating

4.7 同步操练

用户在格林酒店进行网上预订房间操作前,需经过管理系统统一验证(操作员认证和管理员认证都在此页面认证)。此外,采用 AJAX 技术可提供用户满意度。最终效果图如图 4-13 所示。



图 4-13　格林酒店登录功能

项目5　商品展示

5.1　项目引入

商品展示是电子商场不可或缺的功能模块，客户需要购买商品时，都要先查看商品等信息。李明是 Smart On Line 电子商城的一位开发人员，接到的任务是协助小组创建商品展示功能模块。

5.2　项目分析

经过小组讨论和仔细分析，建议 Smart On Line 电子商城站点采用列表展示商品和大图标展示商品两种方式，让客户自由选择合适的商品展示方式，并提供分页浏览功能。大图标方式展示商品信息采用 DataList 控件来实现，并用 PagedDataSource 对象实现分页预览功能。列表方式展示商品信息采用 GridView 控件实现。

5.3　知识准备

5.3.1　DataList 基本知识

DataList 控件用于显示限制于该控件的项目的重复列表，其使用方式和 Repeater 控件相似，也是使用模板标记。DataList 控件可被绑定到数据库表、XML 文件或者其他项目列表，DataList 控件在重复的列表中显示数据项，并且可以选择支持对这些项进行选择和编辑。DataList 中列表项的内容和布局是使用模板定义的。每个 DataList 至少必须定义一个 ItemTemplate；但可以使用多个可选模板自定义列表的外观。表 5-1 描述了这些模板。

表 5-1　DataList 模板项

模板名称	说　明
ItemTemplate	定义列表中各项的内容和布局。为必选项
AlternatingItemTemplate	如果定义，则确定交替项的内容和布局；如果未定义，则使用 ItemTemplate
SeparatorTemplate	如果定义，则呈现在项（以及交替项）之间；如果未定义，则不呈现分隔符
SelectedItemTemplate	如果定义，则确定选定项的内容和布局；如果未定义，则使用 ItemTemplate（AlternatingItemTemplate）

续表

模板名称	说　　明
EditItemTemplate	如果定义，则确定编辑的项的内容和布局；如果未定义，则使用 ItemTemplate（AlternatingItemTemplate 或 SelectedItemTemplate）
HeaderTemplate	如果定义，则确定列表标题的内容和布局；如果未定义，则不呈现标题
FooterTemplate	如果定义，则确定列表页脚的内容和布局；如果未定义，则不呈现页脚

模板定义用于确定项目显示的 HTML 元素和控件，以及这些元素的布局。样式格式设置（字体、颜色和边框属性）是通过样式设置的。每个模板都有自己的样式属性。例如，EditItemTemplate 的样式通过 EditItemStyle 属性设置。第三组属性影响 DataList 的整体呈现。默认情况下，DataList 项在表中作为单独一个垂直的列呈现。将 RepeatLayout 属性设置为 Flow 会从列表的呈现中移除 HTML 表结构。DataList 通过 RepeatDirection 属性支持有方向的呈现，这意味着它可以水平或垂直地呈现其项，见表 5-2 和表 5-3。既然页宽度是 Web 用户界面中开发人员必须控制的维度，DataList 也允许开发人员控制呈现的“列”（RepeatColumns）的数目，无论这些项是水平呈现还是垂直呈现的。如果 RepeatDirection 为 Horizontal 并且 RepeatColumns 为 5，则这些项在包含五个列的行中呈现。

表 5-2　DataList 水平显示样例

1	2	3	4	5
6	7	8	9	10
11	12	13		

如果 RepeatDirection 为 Vertical 并且 RepeatColumns 仍然设置为 5，则这些项分五列呈现，每列的长度等于总项数的 1/5。

表 5-3　DataLidst 垂直显示样例

1	4	7	10	12
2	5	8	11	13
3	6	9		

1. 编辑模板

DataList 是一种显示数据控件，它与 GridView 不同的是，它全部使用模板进行设计，并且 DataList 的模板是对整行设置，而不是像 GridView 那样只对某一列进行模板设计。编辑 DataList 模板可通过单击“智能提示”，再单击“编辑模板”菜单项（见图 5-1），打开“项模板”编辑窗口（见图 5-2）。

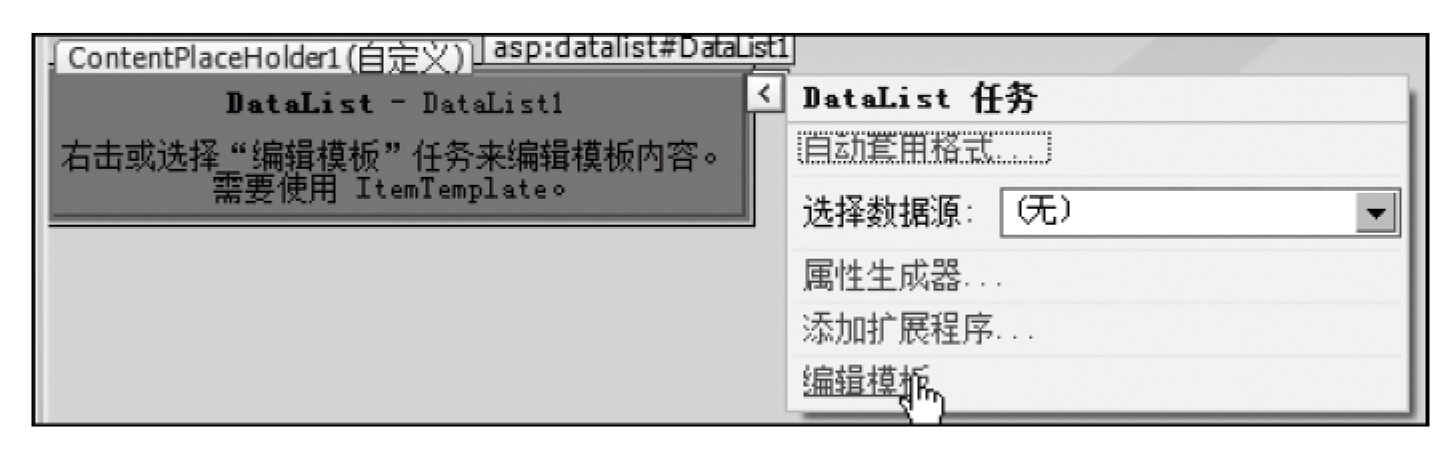


图 5-1　编辑模板



图 5-2 项模板编辑窗口

2. 数据绑定

DataList 控件的默认行为是在 HTML 表格中显示数据库记录。

示例 1：演示如何显示 pubs 数据库 authors 表中的数据。

① 拖动 DataList 控件到 Web 页面上。

② 单击智能提示，单击“编辑模板”，打开“项模板”窗口。拖动 Label 控件到“项模板”窗口中，单击“智能提示”，再单击“编辑 DataBindings”菜单项（如图 5-3 所示），打开“绑定编辑”窗口（见图 5-4）。



图 5-3 编辑 DataBindings

③ 单击 Text 属性，在表达式对话框中输入 Eval("au_fname")，单击“确定”按钮。

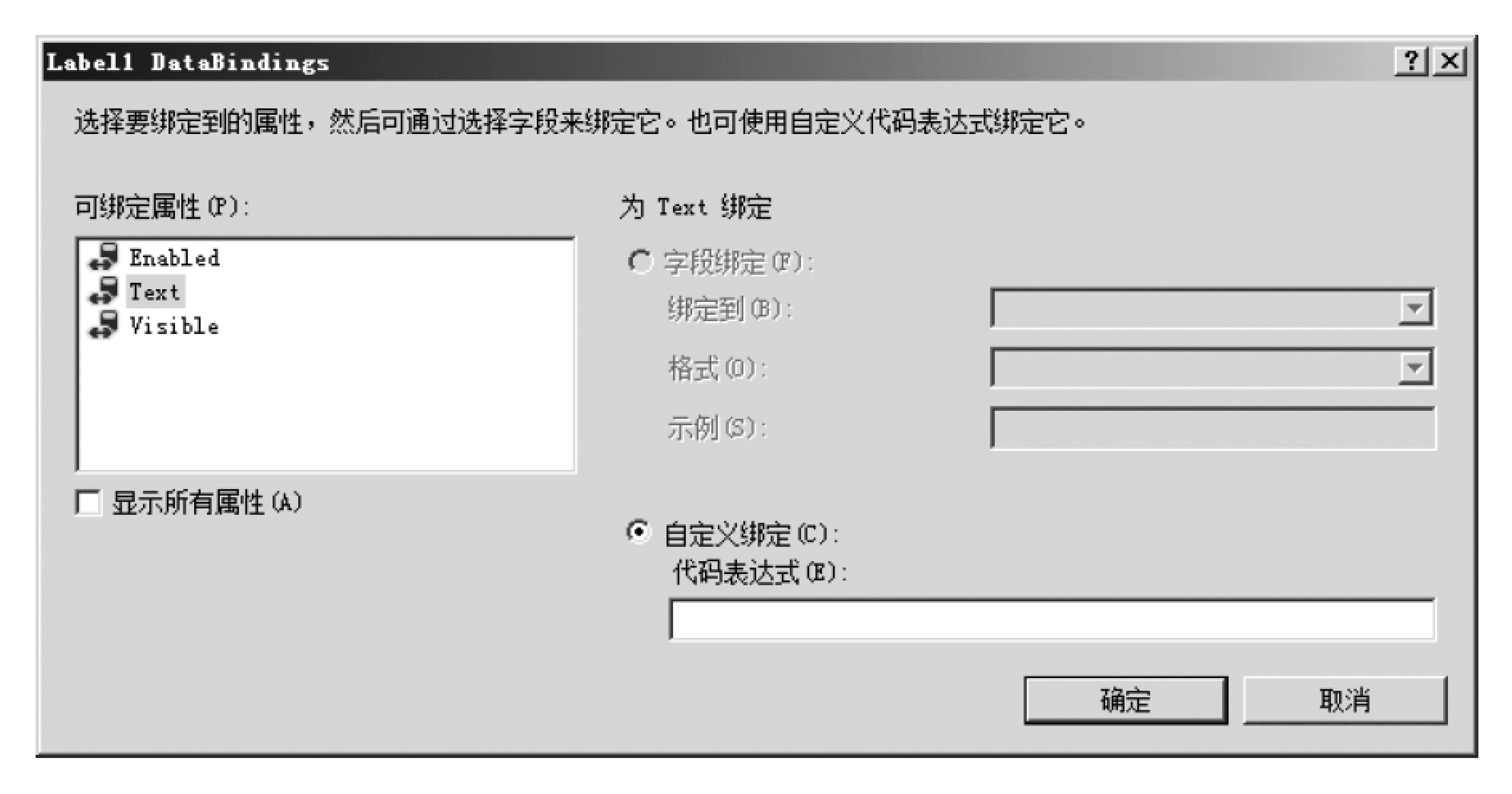


图 5-4 “绑定编辑”窗口

④ 到第三个步骤中已经完成数据显示模板的显示和列绑定。接下来进行 DataList 的数据源的绑定。按功能键 F7，进入后台代码页面，在 Page_Load 页面中输入下列代码，然后按 F5 功能键运行，如图 5-5 所示。

```
SqlConnection conn;
SqlCommand cmd;
SqlDataReader dr;
conn= new SqlConnection("Server= localhost; Database= Pubs;uid= sa;pwd= zzb");
cmd= new SqlCommand("Select au_fname From authors", conn);
```

```
conn.Open();
dr=cmd.ExecuteReader();
DataList1.DataSource=dr;
DataList1.DataBind();
dr.Close();
conn.Close();
```

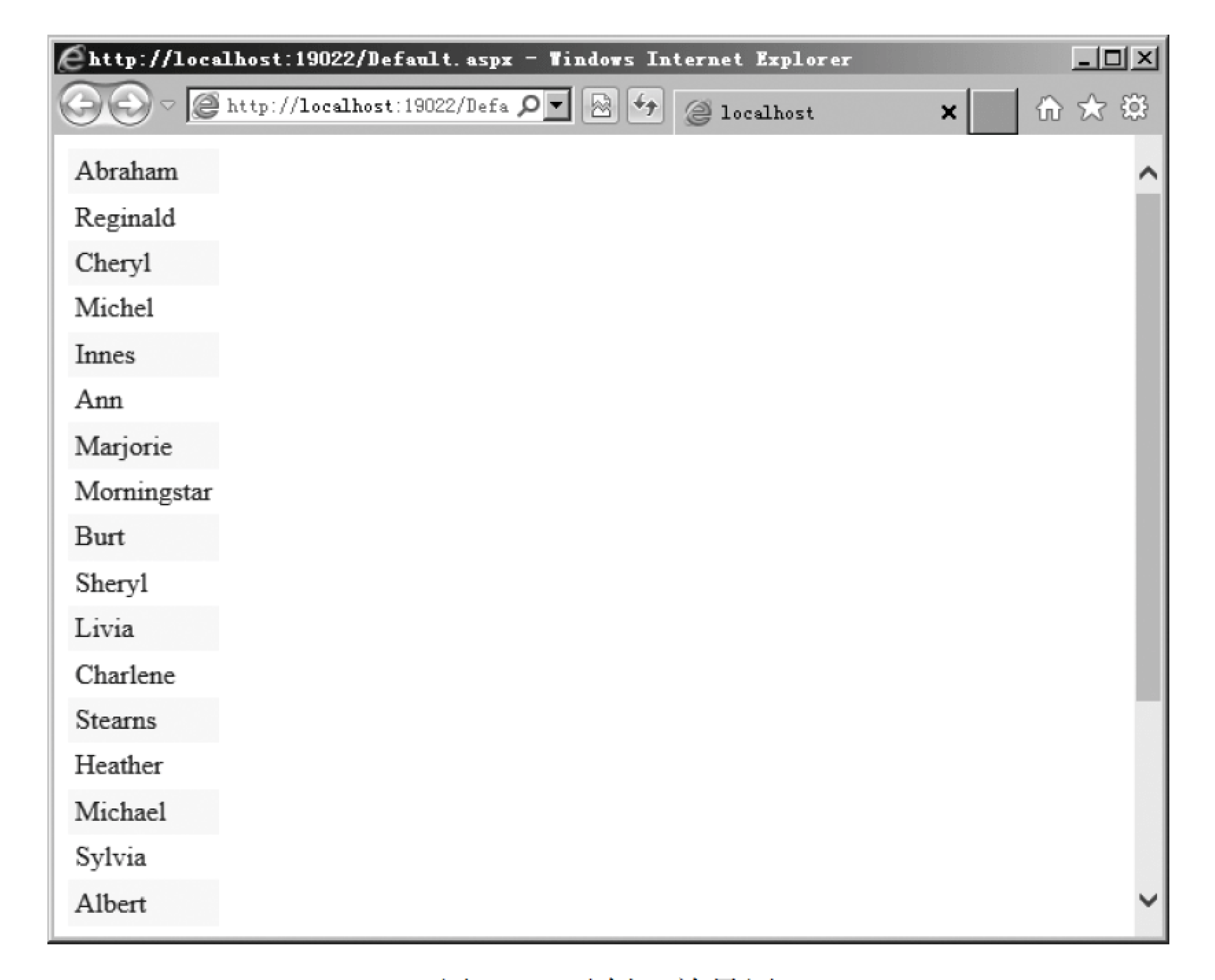


图 5-5　示例 1 效果图

当代码中出现 SqlConnection 等对象不可识别时，可使用 using 指令导入 SqlClient 命名空间。

```
using System.Data.SqlClient;
```

页面代码如下所示。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:DataList ID="DataList1" runat="server" CellPadding="4"
        ForeColor="#333333">
            <AlternatingItemStyle BackColor="White" />
            <FooterStyle BackColor="#990000" Font-Bold="True"
              ForeColor="White" />
            <HeaderStyle BackColor="#990000" Font-Bold="True"
              ForeColor="White" />
```

```
            <ItemStyle BackColor="#FFFBD6" ForeColor="#333333" />
            <ItemTemplate>
                <asp:Label ID="Label1"
                  runat="server" Text='<%# Eval("au_fname") %>'></asp:Label>
            </ItemTemplate>
            <SelectedItemStyle BackColor="#FFCC66" Font-Bold="True"
              ForeColor="Navy" />
        </asp:DataList>

    </div>
    </form>
</body>
</html>
```

3. 捕获 DataList 控件中产生的事件

DataList 控件支持事件冒泡，可以捕获 DataList 内包含的控件产生的事件，并且通过普通的子程序处理这些事件。讲到这里，读者可能不太明白事件冒泡的好处所在，可以反过来思考：如果没有事件冒泡，那么对于 DataList 内包含的每一个控件产生的事件都需要定义一个相应的处理函数，如果 DataList 中包含 10000 个控件呢？或者更多呢？那要写若干个事件处理程序。有了事件冒泡，不管 DataList 中包含多少个控件，只需要一个处理程序就可以了。DataList 控件支持五个事件。

（1）EditCommand：由带有 CommandName＝"edit"的子控件产生。

（2）CancelCommand：由带有 CommandName＝"cancel"的子控件产生。

（3）UpdateCommand：由带有 CommandName＝"update"的子控件产生。

（4）DeleteCommand：由带有 CommandName＝"delete"的子控件产生。

（5）ItemCommand：DataList 的默认事件。

有了这五个事件，那么当单击 DataList 控件中的某一个按钮的时候，应该触发哪一个事件呢？什么时候才触发它们呢？在 ASP. NET 中有三个控件带有 CommandName 属性，分别是 Button、LinkButton 和 ImageButton，可以设置它们的 CommandName 属性来表示容器控件内产生的时间类型。比如，如果设置 DataList 中的一个 LinkButton 的 CommandName 属性为 update，那么单击此按钮的时候，将会触发 DataList 的 CancelCommand 事件，可以将相关处理代码写到对应的事件处理程序中去。

示例 2：演示如何触发 DataList 删除、更新、编辑事件。

① 在示例 1 基础上编辑模板页。在“项模板”编辑窗口中，拖动 ImageButton 到 Label1 控件右边。单击 CommandName 属性，输入 delete(必须是 delete，不能更改为其他，如图 5-6 所示)。再单击 Text 属性，输入 delete。

② 单击 DataList1，双击 DataList1 的 DeleteCommand 事件，如图 5-7 所示。进入代码编辑窗口，输入下列事件相应的代码。该代码的作用是在网页中输出 delete 文字，如图 5-8 所示。

```
protected void DataList1_DeleteCommand1(object source, DataListCommandEventArgs e)
{
    Response.Write("<li>delete");
}
```

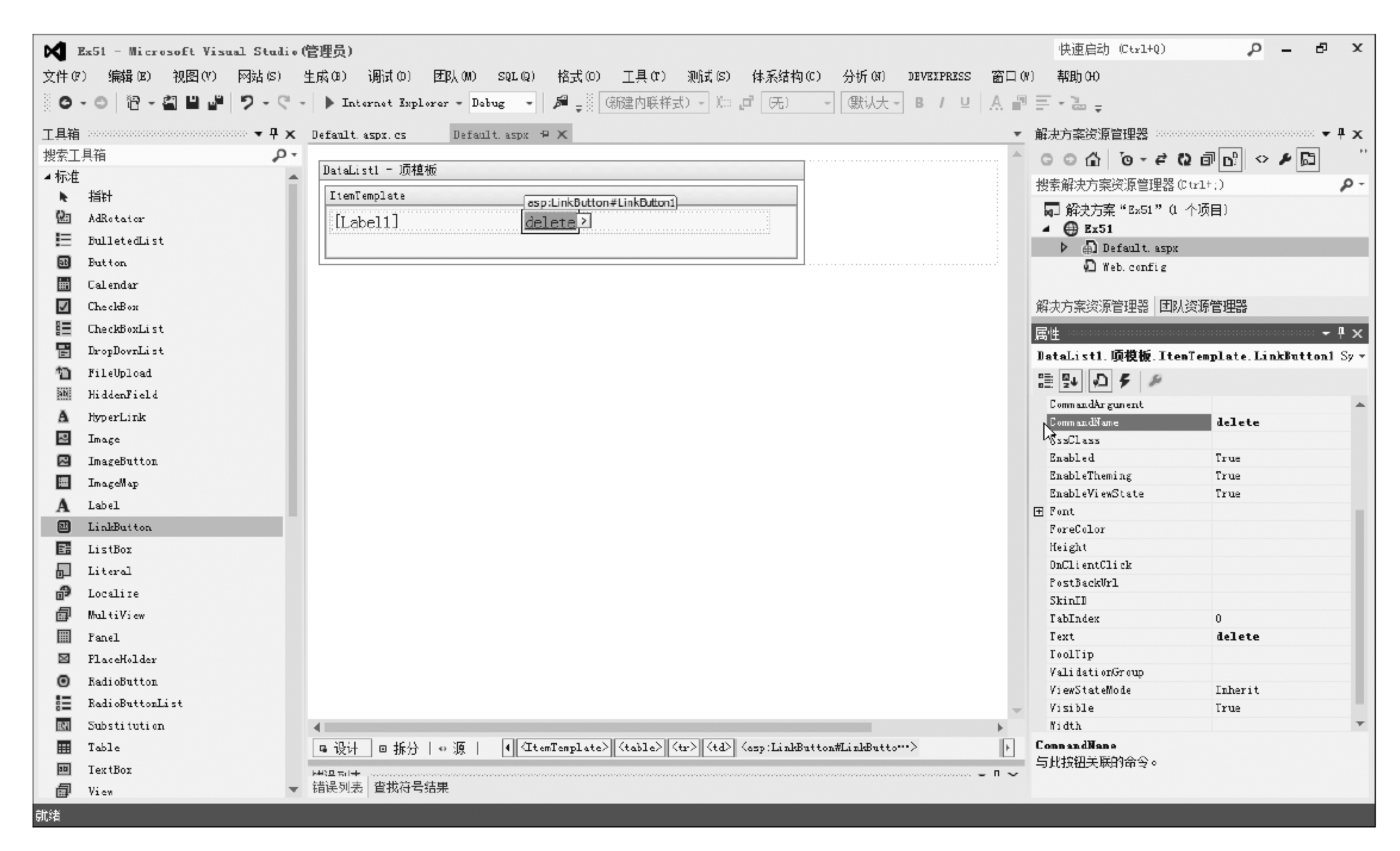


图 5-6　设置 CommandName 属性

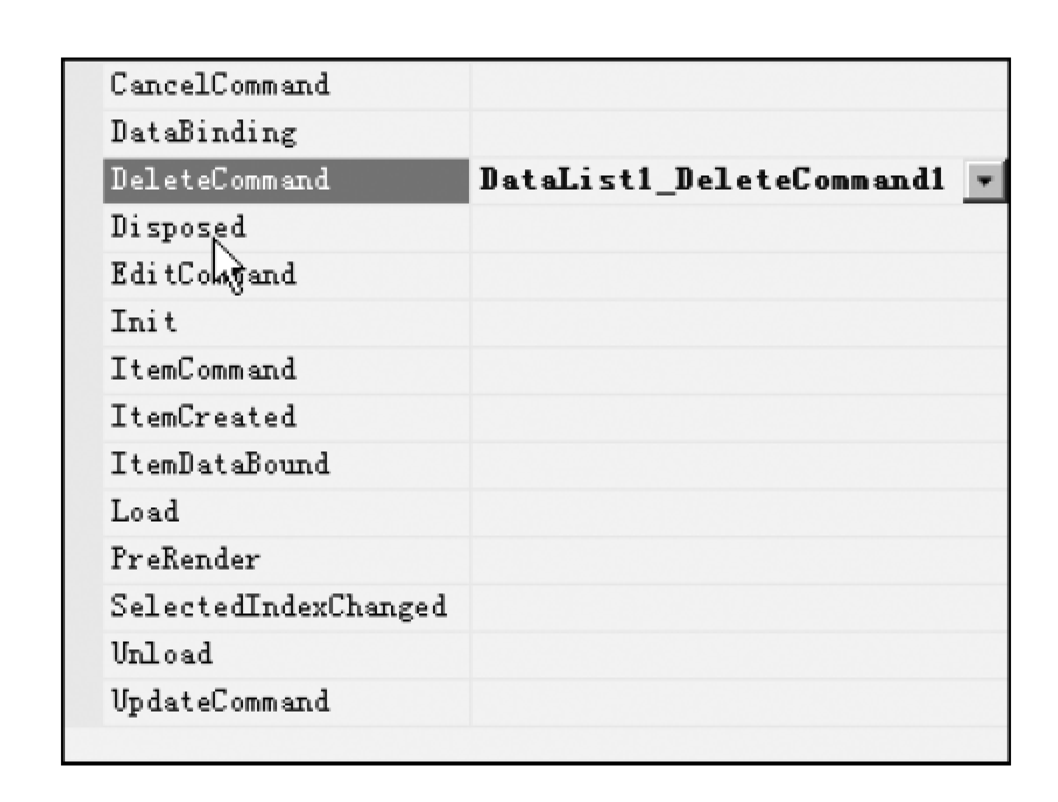


图 5-7　DataList 删除事件

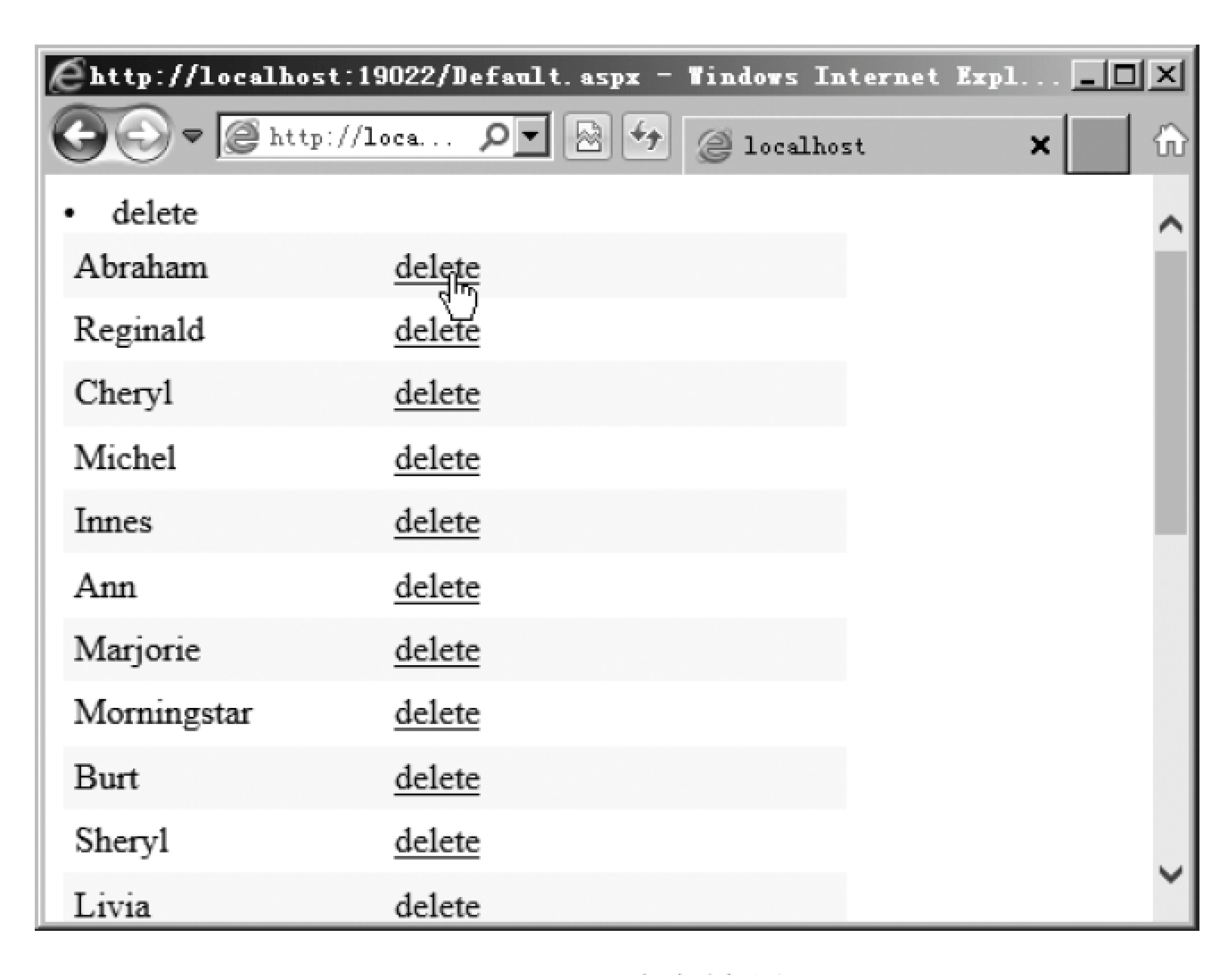


图 5-8　删除效果

③ 用同样的方式创建 update 和 edit 功能，如图 5-9 所示。

Abraham	delete update edit
Reginald	delete update edit
Cheryl	delete update edit
Michel	delete update edit
Innes	delete update edit
Ann	delete update edit
Marjorie	delete update edit
Morningstar	delete update edit
Burt	delete update edit
Sheryl	delete update edit
Livia	delete update edit
Charlene	delete update edit
Stearns	delete update edit
Heather	delete update edit
Michael	delete update edit

图 5-9　示例 2 效果图

在 DataList 中显示的三个 LinkButton 控件分别与相应的程序相关联。当单击名为 delete 的 LinkButton 控件的时候，就触发 DataList 控件 DeleteCommand 事件，该事件与 DataList1_DeleteCommand 函数相关联。读者可能已经注意到与 DataList 关联的函数都带有一个 DataListCommandEventArgs 参数。该参数表示从 DataList 传递给该函数的信息。DataListCommandEventArgs 具有如下属性。

- CommandArgument：表示来自于产生该事件控件的 CommandArgument 属性值。
- CommandName：表示产生该事件的命令名称。
- CommandSource：表示产生该事件的 DataList 控件。
- Item：表示来自 DataList 的项。就是 DataList 中发生事件的那一项。该属性非常有用，在后面的章节中会经常用到。

4. 使用 DataList 控件中的 DataKeys 集合

在选择 DataList 中的一个项时，通常需要获取与这个项相关联的主键的值。可以使用 DataKeys 集合来获取与一个项相关联的主键的值。假设要在 DataList 中显示一个名为 Authors 的数据库表，其中包含两个名为 au_id 和 au_fname 的列，当用户选择 DataList 中的一个项时，要提取与被选项相关联的 au_id 列的值，要实现这个操作，则需要设置 DataList 控件的 DataKeyField 属性：

```
<asp:DataList
ID="DataList1"
DataKeyField="au_id"
OnItemCommand="DataList1_ItemCommand"
Runat="Server">
```

如果把数据库表的主键类的名称赋值给 DataKeyField 属性，那么当绑定 DataList 到 Authors 的数据表时，一个名为 DataKeys 的特殊集合就自动生成了。DataKeys 集合包含来自数据库表的所有的主键值，其顺序与 DataList 中的项相同。

注意：只有当所使用的数据表具有单个主键列时，才可以使用 DataKeys 集合。也就是说不能使用联合主键。

在创建了 DataKeys 集合后，就可以通过传递项的索引值给 DataKeys 集合来获取 DataList 中与相关项关联的主键值。比如，要获取由 DataList 显示的第三项的主键值，就可以使用以下语句：

```
DataList1.DataKeys[2]
```

如果要在 DataList 控件的事件处理函数中发生事件的项的主键值，则可以使用语句：

```
DataList1.DataKeys[e.Item.ItemIndex]
```

5. 编辑 DataList 中的项

DataList 控件具有一个名为 EditItemTemplate 的模板，可以在 EditItemTemplate 中放置表单控件，以便能在 DataList 中编辑特定的项。当 DataList 的 EditItemIndex 属性(该属性默认值为－1，表示不显示 EditItemTemplate 模板)的值为 DataList 某一项的索引的时候，对应的项将会以 EditItemTemplate 模板显示。注意，在 DataList 中一次只能有一个项被选定来编辑。如下程序演示了如何编辑 DataList 的项，其中最主要的部分就是 UpdateCommand 事件的处理程序，因为 DataList 不会自动处理更新数据表的操作，必须自己编写所有的代码。

网页 HTML 代码如下。

```
<html>
  <head>
    <title>DataListEdit.aspx</title>
  </head>
  <body>
    <form runat="Server">
      <asp:datalist id="DataList1"
        datakeyfield="au_id"
        oneditcommand="DataList1_EditCommand"
        oncancelcommand="DataList1_CancelCommand"
        ondeletecommand="DataList1_DeleteCommand"
        onupdatecommand="DataList1_UpdateCommand"
        repeatcolumns="4" gridlines="Both"
        cellpadding="10"
        edititemstyle-backcolor="lightgrey" runat="Server">
        <itemtemplate>
            &lt;%#DataBinder.Eval(Container.DataItem,
            &quot;au_lname&quot;)%&gt;
            - &lt;%#DataBinder.Eval(Container.DataItem, &quot;phone&quot;)%&gt;
        <br />
      <asp:linkbutton text="Edit!" commandname="edit" runat="Server" />
```

```
        </itemtemplate>
        <edititemtemplate>
          <b>Last Name:</b>
          <br />
          <asp:textbox
              id="txtLastName"
              text="&lt;%#DataBinder.Eval(Container.DataItem, &quot;au_lname-
              &quot;)%&gt;"
              runat="Server" />
              <p><b>Phone:</b><br />
          <asp:textbox
              id="txtPhone"
              text="&lt;%#DataBinder.Eval(Container.DataItem, &quot;phone&
              quot;)%&gt;"
              runat="Server" />
              </p><p>
          <asp:linkbutton text="Update!"
              commandname="update" runat="Server" />
          <asp:linkbutton text="Delete!"
              commandname="delete" runat="Server" />
          <asp:linkbutton text="Cancel!"
              commandname="cancel" runat="Server" />        </p>
        </edititemtemplate>
      </asp:datalist>
    </form>
  </body>
</html>
```

后台处理代码如下。

```
void Page_Load(Object sender, EventArgs e)
{
    if (! IsPostBack) {
        BindDataList();
    }
}

void BindDataList() {
    SqlConnection conn;
    SqlCommand cmd;
    SqlDataReader dr;
    conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;
    pwd=123");
    cmd=new SqlCommand("Select au_id, au_lname, phone From Authors Order by
    au_lname", conn);
    conn.Open();
    dr=cmd.ExecuteReader();
    DataList1.DataSource=dr;
    DataList1.DataBind();
    dr.Close();
```

```
        conn.Close();
    }
    void DataList1_EditCommand(object s, DataListCommandEventArgs e) {
        DataList1.EditItemIndex=e.Item.ItemIndex;
        BindDataList();
    }
    void DataList1_CancelCommand(object s, DataListCommandEventArgs e) {
        DataList1.EditItemIndex=-1;
        BindDataList();
    }

    void DataList1_DeleteCommand(object s, DataListCommandEventArgs e) {
        SqlConnection conn;
        string strDelete;
        SqlCommand cmdDelete;
        string strAuthorID;
        strAuthorID=DataList1.DataKeys[(int)e.Item.ItemIndex].ToString();
        conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;
        pwd=123");
        strDelete="Delete Authors Where au_id=@authorID";
        cmdDelete=new SqlCommand(strDelete, conn);
        cmdDelete.Parameters.Add("@authorID", strAuthorID);
        conn.Open();
        cmdDelete.ExecuteNonQuery();
        conn.Close();
        DataList1.EditItemIndex=-1;
        BindDataList();
    }

    void DataList1_UpdateCommand(object s, DataListCommandEventArgs e) {
        SqlConnection conn;
        string strUpdate;
        SqlCommand cmdUpdate;
        string strAuthorID;
        strAuthorID=DataList1.DataKeys[(int)e.Item.ItemIndex].ToString();
        TextBox txtLastName=(TextBox)e.Item.FindControl("txtLastName");
        TextBox txtPhone=(TextBox)e.Item.FindControl("txtPhone");
        conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;
        pwd=123");
        strUpdate="Update Authors set au_lname=@lastname, phone=@phone Where
        au_id=@authorID";
        cmdUpdate=new SqlCommand(strUpdate, conn);
        cmdUpdate.Parameters.Add("@authorID", strAuthorID);
        cmdUpdate.Parameters.Add("@lastname", txtLastName.Text);
        cmdUpdate.Parameters.Add("@phone", txtPhone.Text);
        conn.Open();
        cmdUpdate.ExecuteNonQuery();
        conn.Close();
        DataList1.EditItemIndex=-1;
        BindDataList();
    }
```

运行以上程序，单击其中某一项中的 edit 按钮的时候，将触发 DataList 控件的 EditCommand 事件。通过执行语句 DataList1. EditItemIndex＝e. Item. ItemIndex，将当前项显示为 EditItemTemplate 模板。在 EditItemTemplate 模板中单击 cancel 按钮的时候，将触发 DataList 控件的 CancelCommand 事件，通过执行语句 DataList1. EditItemIndex＝－1，将当前项恢复显示为 ItemTemplate 模板。当单击 update 按钮的时候，将触发 DataList 控件的 UpdateCommand 事件。在 UpdateCommand 事件的处理程序中，需要取得当前项对应的主键值及修改之后的值才能够完成更新，DataList1. DataKeys[(int)e. Item. ItemIndex]. ToString()语句用来获取当前主键值。注意不要遗忘指定 DataList1 的 DataKeyField 属性值为主键字段。DataKeys 集合返回数据类型为 object 类型，还需要调用其 ToString()方法来转换成字符串类型。通过执行语句(TextBox)e. Item. FindControl("txtLastName")来获得更新后的 au_lname 的值，FindControl("控件 ID")方法通过检索当前项中包含的特定 ID 的控件，其返回值类型为 Control 类型，所以还需要强制将其转换为 TextBox 类型，如果没有找到该 ID 的控件，则返回 0，再通过取得的主键值和更新后的字段的值更新数据库表。最后不要忘了重新绑定一下数据，否则数据库里最新的数据将不会显示出来。

5.3.2　DataList 分页

DataList 相对 GridView 控件具有更高的样式自定义性，但是 DataList 没有分页功能，有时很不方便。PagedDataSource 类封装了 GridView 控件的属性，从而使 DataList 控件可以执行分页。它是一个数据的容器，其执行步骤是先把数据从数据库中读取出来放在这个容器中，然后设置容器的属性，并取出当前要显示的页上的部分数据，再将此部分数据绑定到页面的显示控件上。

PageDataSource 类的属性见表 5-4。

表 5-4　PageDataSource 类的属性

PagedDataSource 类的属性	作　　用
AllowCustomPaging	获取或设置指示是否启用自定义分页的值
AllowPaging	获取或设置指示是否启用分页的值
Count	获取要从数据源使用的项数
CurrentPageIndex	获取或设置当前页的索引
DataSource	获取或设置数据源
DataSourceCount	获取数据源中的项数
FirstIndexInPage	获取页中的第一个索引
IsCustomPagingEnabled	获取一个值，该值指示是否启用自定义分页
IsFirstPage	获取一个值，该值指示当前页是否是首页
IsLastPage	获取一个值，该值指示当前页是否是最后一页
IsPagingEnabled	获取一个值，该值指示是否启用分页
IsReadOnly	获取一个值，该值指示数据源是否是只读的
IsSynchronized	获取一个值，该值指示是否同步对数据源的访问(线程安全)
PageCount	获取显示数据源中的所有项所需要的总页数
PageSize	获取或设置要在单页上显示的项数
VirtualCount	获取或设置在使用自定义分页时数据源中的实际项数

示例 3：从 Pubs 的 Jobs 数据表中获取数据，采用分页方式展示数据如图 5-10 所示。当单击“上一页”按钮，则切换到上一页数据页。“下一页”按钮的作用也类似。

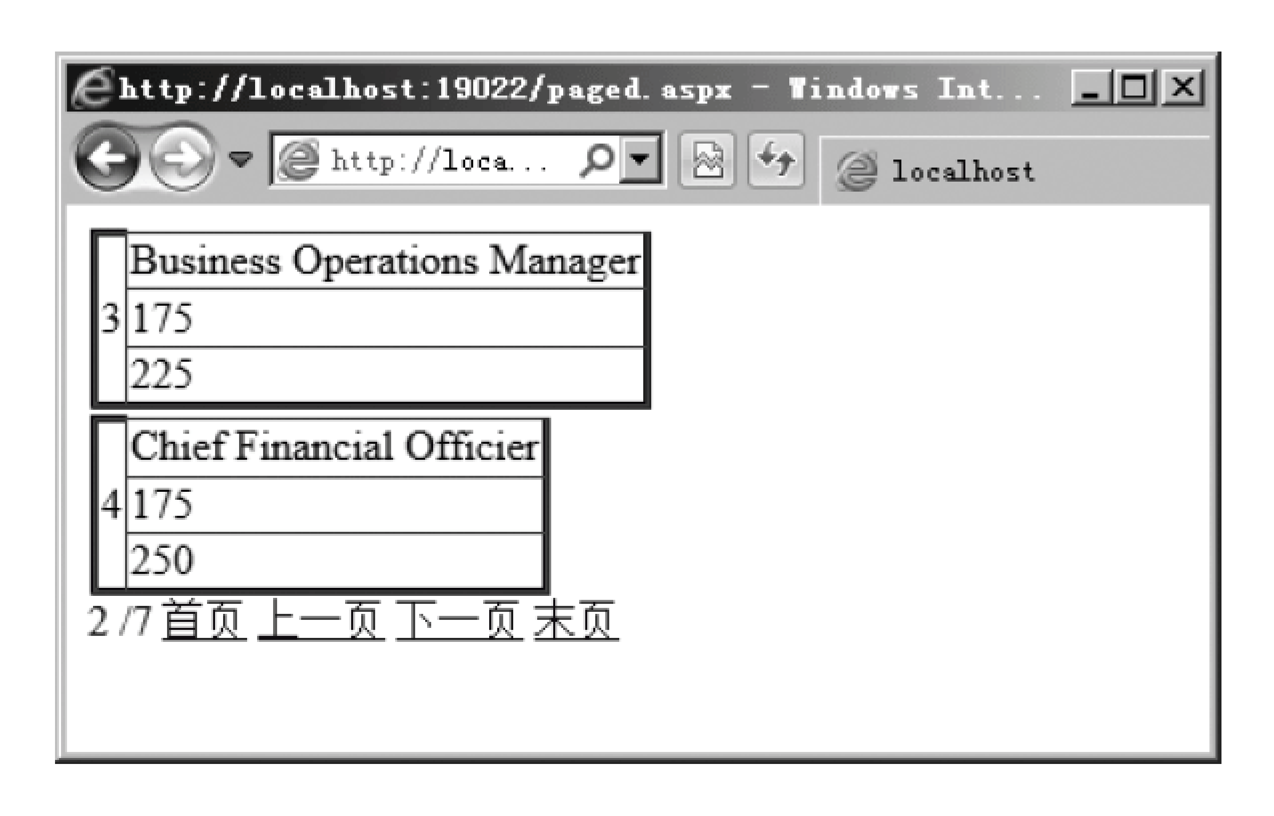


图 5-10　示例 3 效果

① 编写数据绑定过程。为便于复用，编写 getData_Jobs 方法，其作用是从数据库中检索数据，并填充到 DataTable 中。按 F7 功能键，切换到代码页面，输入以下代码。

```
private DataTable getData_Jobs()
{
    string sql="select * from jobs";
    SqlConnection conn;
    SqlCommand cmd;
    SqlDataReader dr;
    conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;
    pwd=zzb");
    cmd=new SqlCommand(sql, conn);
    conn.Open();
    SqlDataAdapter sda=new SqlDataAdapter(cmd);
    DataTable dt=new DataTable();
    sda.Fill(dt);
    conn.Close();
    return dt;
}
```

② 拖动 DataList 控件到页面，单击智能提示按钮，单击“编辑模板”按钮，拖动 4 个 Label 控件按照图 5-11 所示布局，修改相应的 ID 分别为 lblId、lblDesc、lblMin、lblMax。



图 5-11　示例 3 布局

③ 编辑绑定。单击 lblID 标签智能提示，单击“编辑 DataBinds”菜单项，单击 Text 属性，在“代码表达式”文本框中输入 Eval("job_id")，如图 5-12 所示。其他三个标签控件操作类似。lblDesc 标签绑定代码表达式为 Eval("job_desc")，lblMin 标签绑定代码表达式为 Eval("min_lvl")，lblMax 标签绑定代码表达式为 Eval("max_lvl")。

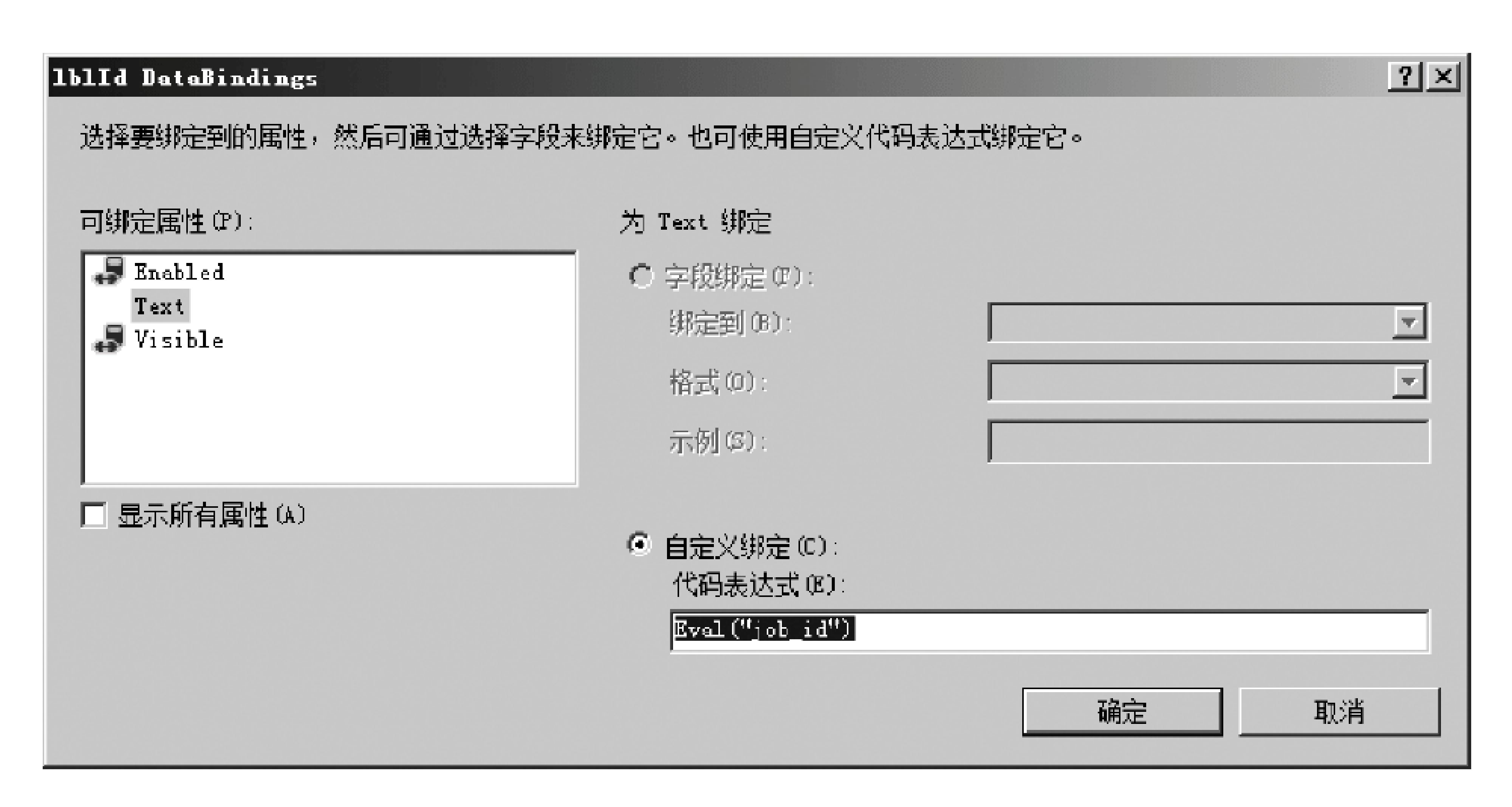


图 5-12　编辑绑定

④ 拖动 Label 和 Button 控件至底部，如图 5-11 所示。更改 Label1 的 ID 为 lblCurrent、Button1 的 ID 为 lbtnFirst、Button2 的 ID 为 lbtnPrev、Button3 的 ID 为 lbtnNext、Button4 的 ID 为 lbtnLast。

⑤ 在代码页中输入 getData 方法，功能是根据传递的页面序号显示相应的数据。

```
Static int i;
private void getData(int pageIndex)
{
    PagedDataSource pd=new PagedDataSource();
    pd.AllowPaging=true;
    pd.PageSize=2;
    pd.CurrentPageIndex=pageIndex - 1;
    pd.DataSource=getData_Jobs().DefaultView;
    DataList1.DataSource=pd;
    DataList1.DataBind();
    lblCurrent.Text=Convert.ToString(pd.CurrentPageIndex+1);
    lblCount.Text="/";
    lblCount.Text+=pd.PageCount.ToString();
    i=pd.PageCount;
    if(pd.IsFirstPage)
    {
        lbtnFirst.Visible=false;
        lbtnPrev.Visible=false;
    }
    else
    {
        lbtnFirst.Visible=true;
        lbtnPrev.Visible=true;
    }
    if(pd.IsLastPage)
```

```
        {
            lbtnLast.Visible=false;
            lbtnNext.Visible=false;
        }
        else
        {
            lbtnLast.Visible=true;
            lbtnNext.Visible=true;
        }
    }
```

这段代码主要运用了 PagedDataSource 对象，将其对象的 DataSource 设置为获取的表的默认视图。然后分别设置 PagedDataSource 对象的 AllowPaging 等属性。变量 i 标识为当前页序号。

⑥ 双击 lbtnFirst 按钮，进入事件相应代码页面，输入如下代码。

```
protected void lbtnFirst_Command(object sender, CommandEventArgs e)
{
    switch(e.CommandName)
    {
        case "First":
            getData(1);
            break;
        case "Next":
            getData(Convert.ToInt32(lblCurrent.Text)+1);
            break;
        case "Prev":
            getData(Convert.ToInt32(lblCurrent.Text) - 1);
            break;
        case "Last":
            getData(i);
            break;
        default:
            getData(1);
            break;
    }
}
```

⑦ 单击 lbtnPrev 按钮，在属性窗口中单击“事件”按钮，单击 Command 属性，单击右侧下拉列表，再单击 lbtnFirst_Command，如图 5-13 所示。其他的 lbtnNext、lbtnLast 也同样设置该事件。

⑧ 设置 DataList 控件样式。单击智能提示，单击“自动套用格式”菜单项，如图 5-14 所示。

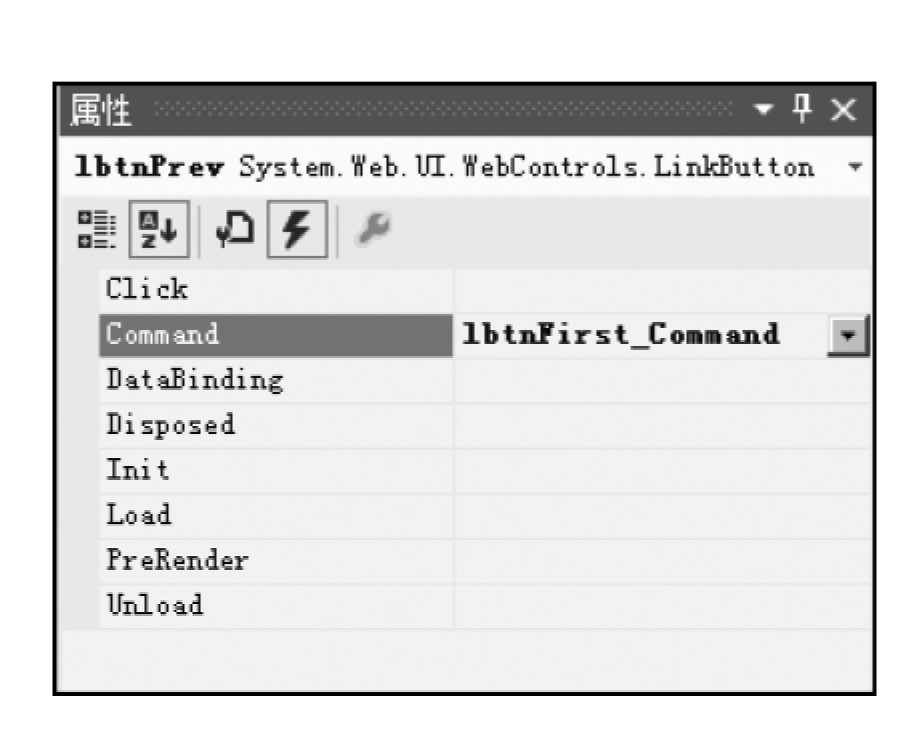


图 5-13　设置 Command 事件



图 5-14　自动套用格式

5.3.3　GridView

GridView 控件为开发人员提供了强大的管理方案,同样 GridView 也支持内置格式,单击“自动套用格式”按钮可以选择 GridView 中的默认格式,如图 5-15 所示。GridView 是以表格为表现形式,GridView 包括行和列,通过配置相应的属性能够编辑相应的行的样式,同样也可以选择“编辑列”选项来编写相应的列的样式,如图 5-16 所示。

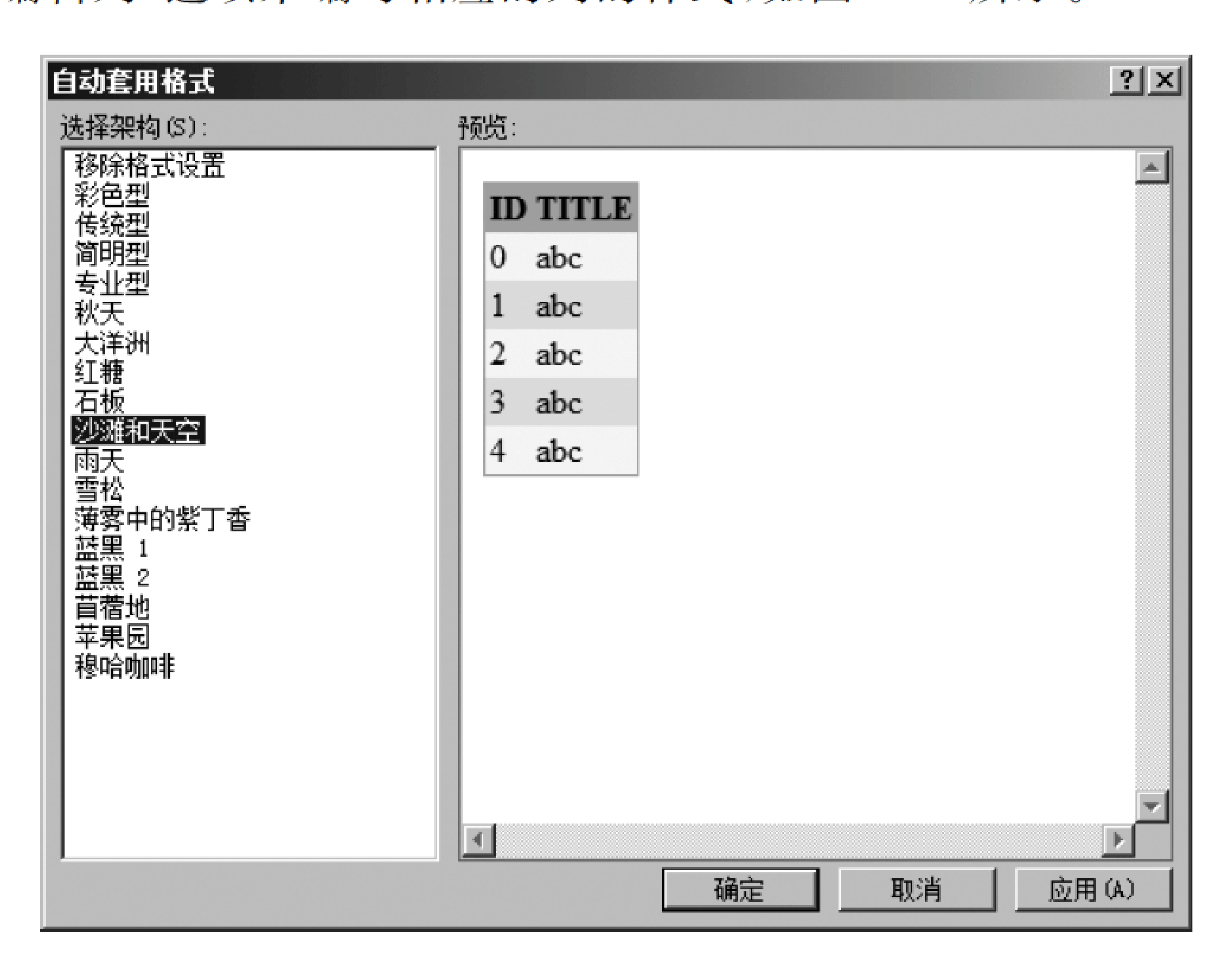


图 5-15　自动套用格式

1. GridView 数据显示

GridView 大部分场合下都是用来绑定数据源,进行数据的显示。一般情况下,可以绑定到 SqlDataSource 控件、DataTable 对象、DataView 对象,也可以绑定到列表对象。常见

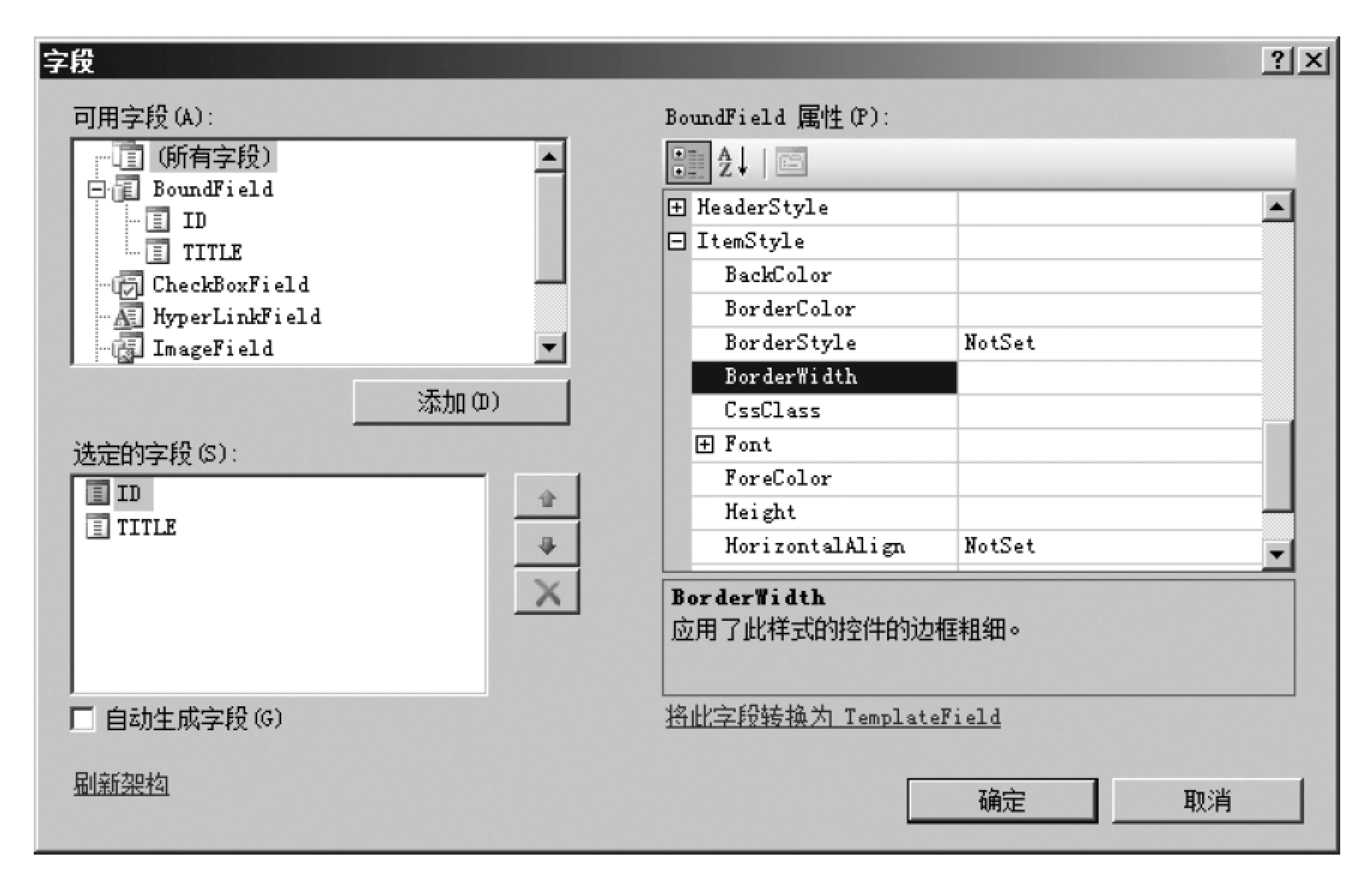


图 5-16 编辑列

的是使用 DataSource 属性进行数据绑定，此选项能够绑定到包括 ADO. NET 数据集和数据读取器在内的各种对象，能够自由实现各种功能，但此方法需要为所有附加功能(如排序、分页和更新)编写代码。特别值得提醒的是，采用 DataSource 属性实现数据显示，除了设置 DataSource 属性外，还必须调用 Gridview 控件的 DataBind 方法，才能最终将数据显示到 Gridview 中。

示例 4：在 GridView 中显示 Pubs 数据库中的 jobs 数据。

① 拖动 GridView 控件到页面。

② 单击“自动套用格式”菜单项，打开“自动套用格式”对话框，单击“彩色型”列表项，单击“确定”按钮，如图 5-17 所示。



图 5-17 “自动套用格式”对话框

③ 按 F7 功能键切换到代码页面，在 Page_Load 事件中输入下面代码，按 F5 功能键运行，得到如图 5-18 所示的结果。

```
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {
        string sql="select * from jobs";
        SqlConnection conn;
        SqlCommand cmd;
        SqlDataReader dr;
        conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;pwd=zzb");
        cmd=new SqlCommand(sql, conn);
        conn.Open();
        SqlDataAdapter sda=new SqlDataAdapter(cmd);
        DataTable dt=new DataTable();
        sda.Fill(dt);
        conn.Close();
        GridView1.DataSource=dt;
        GridView1.DataBind();
    }
}
```

job_id	job_desc	min_lvl	max_lvl
1	New Hire - Job not specified	10	10
2	Chief Executive Officer	200	250
3	Business Operations Manager	175	225
4	Chief Financial Officier	175	250
5	Publisher	150	250
6	Managing Editor	140	225
7	Marketing Manager	120	200
8	Public Relations Manager	100	175
9	Acquisitions Manager	75	175
10	Productions Manager	75	165
11	Operations Manager	75	150
12	Editor	25	100
13	Sales Representative	25	100
14	Designer	25	100

图 5-18 示例 4 效果图

2. GridView 七种字段

ASP. NET 中 GridView 绑定到数据源时，可以自动显示数据源的各个字段。只要设定其 AutoGenerateColumns 为 TRUE 即可。但自动显示有其不好的一面，因为不能自定义控制显示的样式。解决以上问题的办法就是指定需要 GridView 显示的字段，GridView 控件支持以下七种类型的 Field。

- BoundField：将数据项显示为文本。
- CheckBoxField：将数据项显示为复选框。
- CommandField：使用链接来支持编辑、删除或选中行。
- ButtonField：将数据项显示为按钮(ImageButton、LinkButton、Button)。
- HyperLinkField：将数据项显示为超链接。
- ImageField：将数据项显示为图片。
- TemplateField：自定义数据项的外观。

① BoundField。GridView 在显示状态时，BoundField 总是直接把数据项显示为文本；GridView 在编辑状态时，BoundFiled 将数据项显示为一个单行的文本框。除了其父类 DataControlField 的几个属性外，还有以下几个重要属性。

- DataField：显示的字段。
- DataFormatString：字段格式化。
- HtmlEncode/HtmlEncodeFormatString：获取或设置一个值，该值指示在 BoundField 对象中显示字段值之前，是否对这些字段值进行 HTML 编码。

注意：FormatString 经常用来格式化数字、日期、字符串、自定义类型。

② CheckBoxField。CheckBoxField 会在相应的列内显示一个复选框，在没有进入编辑模式时，其复选框处于禁用状态。CheckBoxField 通过设置 Text 属性，可以在每一个复选框旁边显示一个标题。

③ CommandField。使用 CommandField 可以定制 GridView 控件中 Edit、Delete、Update、Cancel、Select 等按钮的外观。需要使用 CommonField 时，不要启用 GridView 的 AutoGenerateEditButton 和 AutoGenerateDeleteButton 属性，因为可以直接使用 CommandField。

CommandField 的一些属性如下。

- ButtonType：指定 Button 类型，可以有 Button、Image、Link 类型。
- CancelText/CancelImageUrl：Cancel 按钮中的文本/图像 URL。
- DeleteText/DeleteImageUrl：Delete 按钮中的文本/图像 URL。
- InsertText/InsertImageUrl：Insert 按钮中的文本/图像 URL。
- EditText/EditImageUrl：Edit 按钮中的文本/图像 URL。
- UpdateText/UpdateImageUrl：Update 按钮中的文本/图像 URL。
- SelectText/SelectImageUrl：Select 按钮中的文本/图像 URL。
- NewText/NewImageUrl：New 按钮中的文本/图像 URL。
- CauseValidation：单击按钮时是否启用校验。
- ValidationGroup：指定和编辑按钮相关验证控件组的名称。

④ ButtonField。使用 ButtonField 可以在 GridView 中显示一个按钮，使用它可以完成自定义或标准的编辑命令。单击 GridView 中的 ButtonField 字段，会触发 GridView 中的 OnRowCommand 事件。可以在这个事件中处理相关的命令事件。

ButtonField 有以下几个属性。

- ButtonType：Button 类型，可以为 Button、Image、LinkButton。
- CauseValidation：指定按钮单击时是否引发验证。
- CommandName：指定 ButtonField 关联的标准编辑命令，可以为 Delete、Edit、Update、Cancel，也可以自定义。
- DataTextField/DataTextFormatString：指定按钮文本的数据项/数据项格式。
- Text：按钮文本。
- ValidationGroup：与按钮相关验证控件组的名称。

⑤ HyperLinkField。HyperLinkField 用来链接到其他页面。当创建两个主从表单的时候，HyperLinkField 非常有用。

HyperLinkField 具有以下属性。

- DataNavigateUrlFields：在 DataNavigateFormatString 中使用的列名称。
- DataNavigateFormatString：格式化链接字符串。
- DataTextField/DataTextFormatString：超链接文本/超链接文本格式化。
- NavigateUrl：链接到其他页面的 URL。
- Target：链接目标，可以使用_blank、_parent、_self、_top。
- Text：超链接的文本。

⑥ ImageField。ImageFIeld 用来显示保存在服务器上的图片，不能用 ImageField 来显示保存在数据库上的图片。

ImageField 有以下几个属性。

- AlternateText：预备文本。
- DataAlternateTextField：使用指定列的值作为预备文本。
- DataAlternateTextFormatString：预备文本格式字符串。
- DataImageUrlField：存放图片路径的列名。
- DataImageUrlFormatString：图片路径格式字符串。
- NullImageUrl：指定预备图片

⑦ TemplateField。使用 TemplateField 可以在 GridView 控件的数据列中添加任何内容，例如 HTML、数据绑定表达式或者 ASP. NET 控件等。可以使用 TemplateField 定制用户界面或者给被编辑字段添加验证。

TempateField 支持以下 6 种类型的模板。

- ALternatingItemTemplate：间隔行模板。
- EditItemTemlpate：编辑行模板。
- FooterTemplate：脚注模板。
- HeaderTemplate：标题模板。
- InsertItemTemplate：插入行模板(不支持 GridView 控件)。
- ItemTemplate：每个显示行模板

示例 5：将示例 4 中的 GridView 标题设置为中文表示。GridView 数据绑定时会自动产生列，需要将自动产生列功能取消，手动使用 BoundField 进行绑定。

① 单击智能提示，单击“编辑列”菜单项，打开“字段”对话框，如图 5-19 所示。单击取消选中“自动生成字段”复选框。

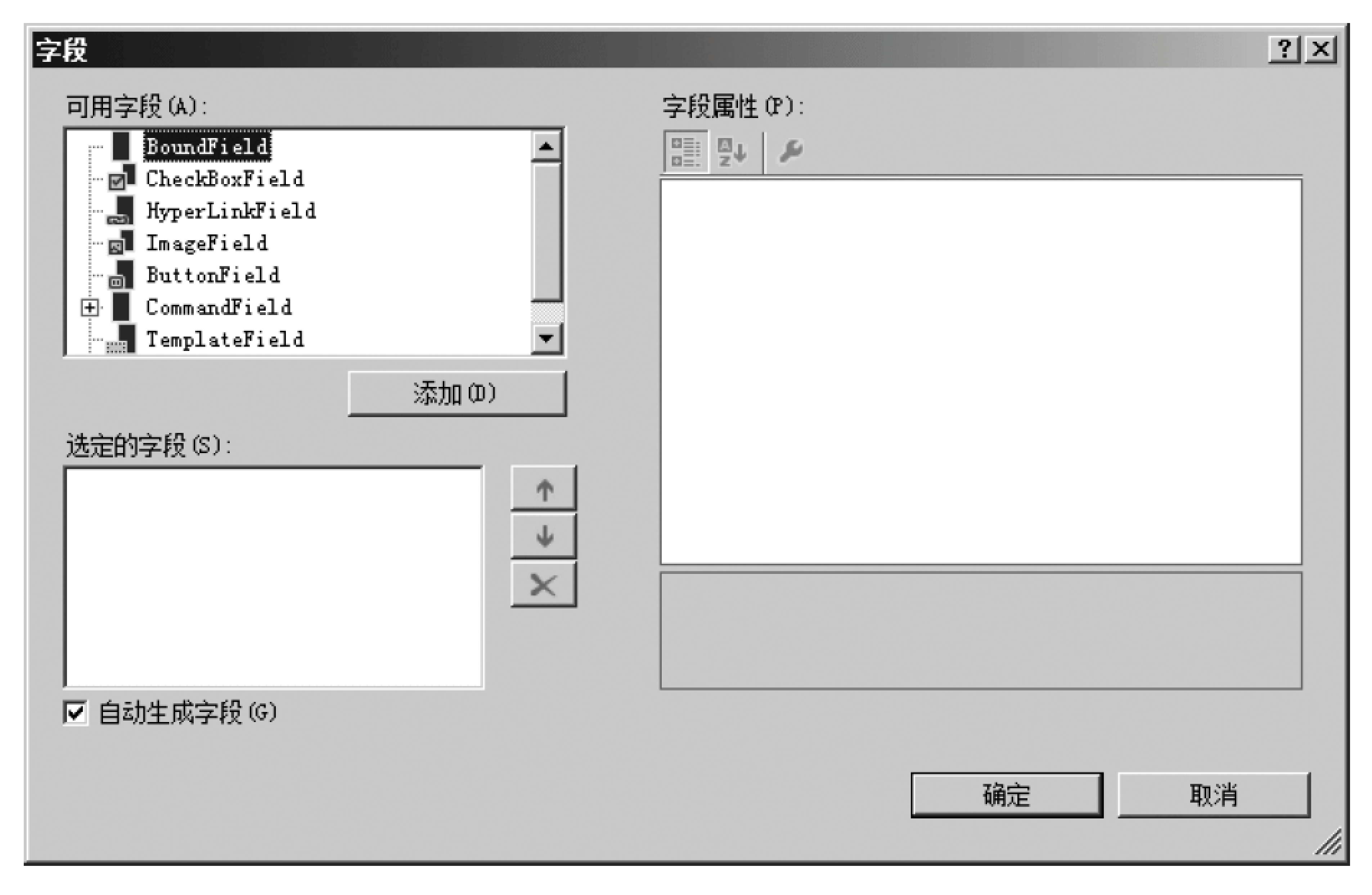


图 5-19 “字段”对话框

② 单击 BoundField，单击“添加”按钮，单击 DataField 属性，输入 job_id。单击 HeaderText 属性，输入“工种编号”，这样 job_id 数据字段就绑定成功了。这里 HeaderText 属性用于显示字段标题，如图 5-20 所示。用同样方式采用 BoundField 绑定其他列的数据，最终效果图如图 5-21 所示。

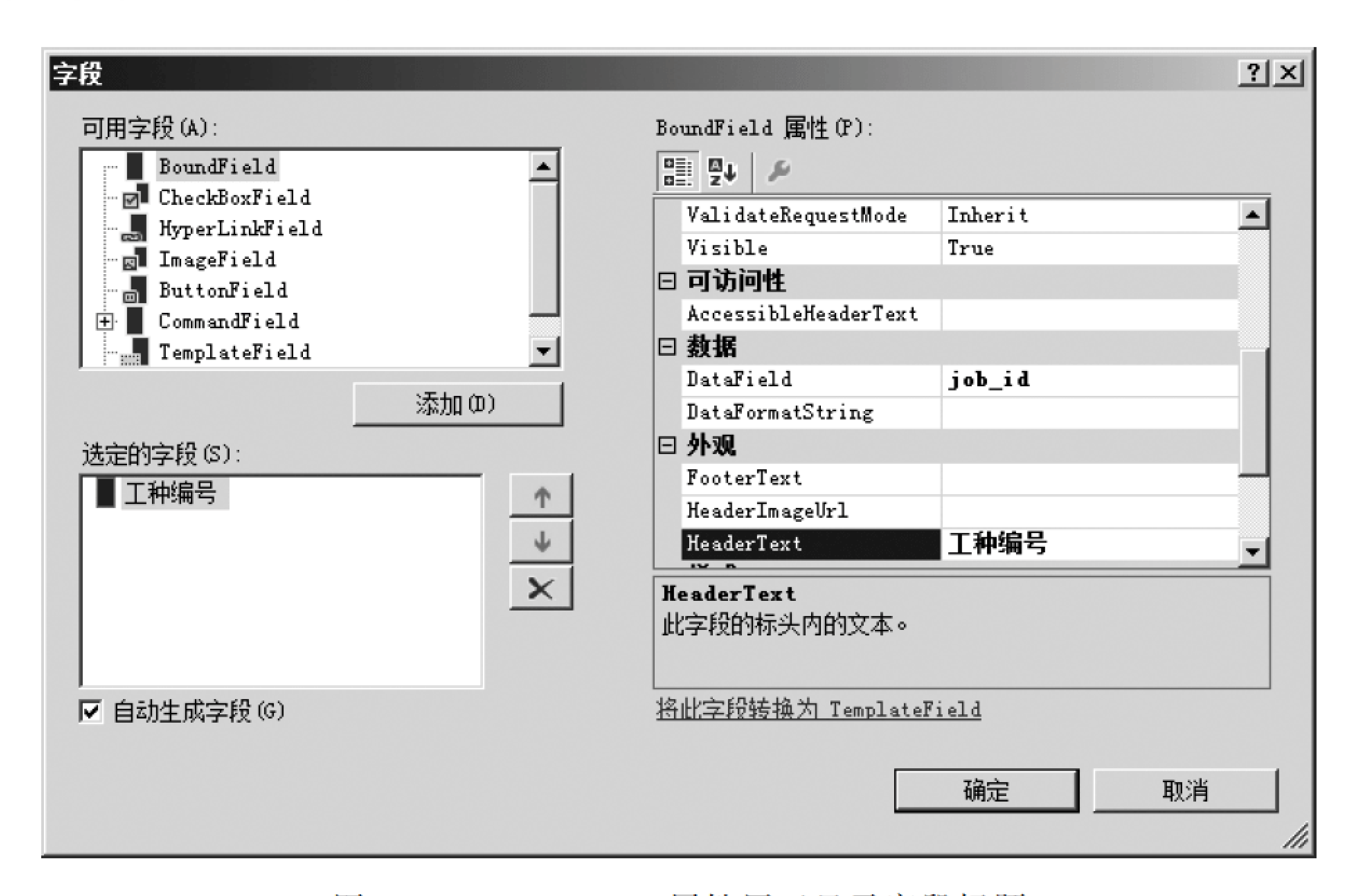


图 5-20 HeaderText 属性用于显示字段标题

示例 6：单击“查看该工种员工信息”链接(见图 5-22)，跳转到显示该工种的员工信息(见图 5-23)。

工种编号	工种描述	最低工资	最高工资
1	New Hire - Job not specified	10	10
2	Chief Executive Officer	200	250
3	Business Operations Manager	175	225
4	Chief Financial Officier	175	250
5	Publisher	150	250
6	Managing Editor	140	225
7	Marketing Manager	120	200
8	Public Relations Manager	100	175
9	Acquisitions Manager	75	175
10	Productions Manager	75	165
11	Operations Manager	75	150
12	Editor	25	100
13	Sales Representative	25	100
14	Designer	25	100

图 5-21　示例 5 效果图

工种编号	工种描述	最低工资	最高工资	
1	New Hire - Job not specified	10	10	查看该工种员工信息
2	Chief Executive Officer	200	250	查看该工种员工信息
3	Business Operations Manager	175	225	查看该工种员工信息
4	Chief Financial Officier	175	250	查看该工种员工信息
5	Publisher	150	250	查看该工种员工信息
6	Managing Editor	140	225	查看该工种员工信息
7	Marketing Manager	120	200	查看该工种员工信息
8	Public Relations Manager	100	175	查看该工种员工信息
9	Acquisitions Manager	75	175	查看该工种员工信息
10	Productions Manager	75	165	查看该工种员工信息
11	Operations Manager	75	150	查看该工种员工信息
12	Editor	25	100	查看该工种员工信息
13	Sales Representative	25	100	查看该工种员工信息
14	Designer	25	100	查看该工种员工信息

图 5-22　超链接列

emp_id	fname	minit	lname	job_id	job_lvl	pub_id	hire_date
KJJ92907F	Karla	J	Jablonski	9	170	9999	1994/3/11 0:00:00
M-R38834F	Martine		Rance	9	75	0877	1992/2/5 0:00:00
MAS70474F	Margaret	A	Smith	9	78	1389	1988/9/29 0:00:00
GHT50241M	Gary	H	Thomas	9	170	0736	1988/8/9 0:00:00

图 5-23　跳转

① 在示例 5 的基础上，进一步添加超链接列。在“字段”对话框中单击 HyperLinkField 列表项，单击“添加”按钮。在右边的属性窗口中单击 DataNavigateUrlFields 属性，输入 job_id。在 DataNavigateUrlFormatString 中输入～/Employees. aspx?jid＝{0}。{0}表示占位符，将 DataNavigateUrlFields 中的第一个字段填充到{0}中，表示 Url 传值。

② 双击打开 Employees. aspx 页面，按 F7 功能键输入下面的代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {
        int jid=Convert.ToInt32(Request.QueryString["jid"].ToString());
        SqlConnection conn;
        SqlCommand cmd;
        SqlDataReader dr;
        conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;
        pwd=zzb");
        cmd=new SqlCommand("Select * From Employee where job_id="+jid, conn);
        conn.Open();
        dr=cmd.ExecuteReader();
        GridView1.DataSource=dr;
        GridView1.DataBind();
        dr.Close();
        conn.Close();
    }
}
```

示例 7：单击“删除”按钮，删除对应的工种，如图 5-24 所示。删除功能可以通过 CommandField 的删除命令来完成。

工种编号	工种描述	最低工资	最高工资		
1	New Hire - Job not specified	10	10	查看该工种员工信息	删除
2	Chief Executive Officer	200	250	查看该工种员工信息	删除
3	Business Operations Manager	175	225	查看该工种员工信息	删除
4	Chief Financial Officier	175	250	查看该工种员工信息	删除
5	Publisher	150	250	查看该工种员工信息	删除
6	Managing Editor	140	225	查看该工种员工信息	删除
7	Marketing Manager	120	200	查看该工种员工信息	删除
8	Public Relations Manager	100	175	查看该工种员工信息	删除
9	Acquisitions Manager	75	175	查看该工种员工信息	删除
10	Productions Manager	75	165	查看该工种员工信息	删除
11	Operations Manager	75	150	查看该工种员工信息	删除
12	Editor	25	100	查看该工种员工信息	删除
13	Sales Representative	25	100	查看该工种员工信息	删除
14	Designer	25	100	查看该工种员工信息	删除

图 5-24　删除功能

① 在“字段”窗口对话框中单击 CommandField 并选择“删除”列表项，单击“添加”按钮。

② 单击 GridView 控件，单击 DataKeyNames 属性，输入 job_id。DataKeyNames 表示主键的列名，可以通过 GridViewEntity. DataKeys[RowIndex]["ColumsName"]来获取它的值。这个步骤切勿跳过。

③ 单击 GridView 控件，单击事件按钮，双击 RowDeleting 事件，进入代码编辑状态。RowDeleting 事件在“删除”按钮被触发时调用。

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    String jid=GridView1.DataKeys[e.RowIndex].ToString();
    SqlConnection conn;
    SqlCommand cmd;
    string sql="delete from jobs where job_id="+jid;
    conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;
    pwd=zzb");
    cmd=new SqlCommand(sql, conn);
    conn.Open();
    cmd.ExecuteNonQuery();
    conn.Close();
}
```

示例 8：在示例 7 的基础上添加分页功能，如图 5-25 所示。

工种编号	工种描述	最低工资	最高工资	
6	Managing Editor	140	225	查看该工种员工信息 删除
7	Marketing Manager	120	200	查看该工种员工信息 删除
8	Public Relations Manager	100	175	查看该工种员工信息 删除
9	Acquisitions Manager	75	175	查看该工种员工信息 删除
10	Productions Manager	75	165	查看该工种员工信息 删除
		123		

图 5-25　分页功能

① 单击 GridView 控件，单击 AllowPaging 属性并将其设为 True。

② 单击 PageSize 属性，输入 5。

③ 单击 GridView 控件，单击事件按钮，双击 PageIndexChanging 事件。输入以下代码。

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    GridView1.PageIndex=e.NewPageIndex;
    Bind();
}
```

其中 Bind 方法是简单的数据查询功能。为便于多次调用，将其封装成 Bind 方法。

```
public void Bind()
{
    string sql="select * from jobs";
    SqlConnection conn;
    SqlCommand cmd;
    SqlDataReader dr;
    conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;pwd=zzb");
    cmd=new SqlCommand(sql, conn);
    conn.Open();
    SqlDataAdapter sda=new SqlDataAdapter(cmd);
    DataTable dt=new DataTable();
    sda.Fill(dt);
```

```
        conn.Close();
        GridView1.DataSource=dt;
        GridView1.DataBind();
    }
```

5.3.4 DetailsView

前面内容介绍了如何使用 TemplateField 来自定义 GridView 和 DetailsView 的输入。TemplateField 使我们可以高度自主地定义某个特定的列，但不管是 GridView 还是 DetailsView，都会有点太规则了，简单地说就是它们都有着四四方方的格子一样的外观。很多情况下这样的格子一样的外观是很不错的，不过有的时候我们却需要使用一个不规则的显示外观。当需要显示一个单独的记录时，使用 FormView 控件就可以实现这种比较随意的外观呈现。与 DetailsView 不同，FormView 并不是由那些杂七杂八的列所组成的。不能给一个 FormView 添加 BoundField 或是 TemplateField，不过 FormView 是使用模板来呈现的。我们可以这样来理解 FormView：把它当作只含有一个 TemplateField 的 DetailsView 控件。FormView 支持以下这些模板。

- ItemTemplate：用于在 FormView 种呈现一个特殊的记录。
- HeaderTemplate：用于指定一个可选的页眉行。
- FooterTemplate：用于指定一个可选的页脚行。
- EmptyDataTemplate：当 FormView 的 DataSource 缺少记录的时候，EmptyDataTemplate 将会代替 ItemTemplate 来生成控件的标记语言。
- PagerTemplate：如果 FormView 启用了分页，这个模板可以用于自定义分页的界面。
- EditItemTemplate/InsertItemTemplate：如果 FormView 支持编辑或插入功能，那么这两种模板可以用于自定义相关的界面。

鉴于 FormView 也是基于模板操作，同 DataList 操作非常类似，本书就不再一一细述。

5.4 项目实施

5.4.1 任务 1：大图标方式显示商品信息

1. 任务目标

(1) 能熟练使用 DataList 显示信息。

(2) 能使用 PagedDataSource 进行分页。

(3) 能使用 ADO.NET 访问数据库。

2. 任务内容

(1) 以大图标方式显示商品信息。

(2) 以分页形式显示商品信息。

3. 任务实施步骤

该任务主要考察使用 DataList 控件的熟练程度。DataList 是基于模板的，而且默认是不能分

页的。这个任务要求掌握 PagedDataSource 对象与 DataList 的联系,具体效果见图 5-26。

图 5-26　以大图标方式显示商品信息

① 创建以大图标方式显示商品信息的内容页。右击 Shop 文件夹,选择"添加"→"添加新项"命令,打开"添加新项"对话框,单击"Web 页面"列表项,文件名文本框中输入 WareListView.aspx。单击选中"选择母版页"复选框。打开"选择母版页"对话框,选择 Layout.master,单击"添加"按钮。

② 使用 ADO.NET 编程检索数据库,将结果保存到 DataTable 对象中。任务中仍旧采用 ADO.NET 编程检索数据库,将检索的过程封装在 getData_Ware 方法中,代码如下。

```
private DataTable getData_Wares()
{
    string sql="select * from T_Ware";
    SqlConnection conn;
    SqlCommand cmd;
    conn=sh.getConn();
    conn.Open();
    cmd=new SqlCommand(sql, conn);
    SqlDataAdapter sda=new SqlDataAdapter(cmd);
    DataTable dt=new DataTable();
    sda.Fill(dt);
    conn.Close();
    return dt;
}
```

③ 使用 DataList 自定义显示方式(见图 5-27)。拖动 DataList 控件到页面,单击"智能提示",单击"编辑模板"菜单项,打开"项模板"编辑窗口。拖动 HTML 标签中的 Table 控件到模板中,将其设置为 6 行 3 列。右击 Table 的列头,选择"修改"→"合并单元格"命令。拖动 Image 控件到第 1 行第 1 列,单击 Image 控件"智能提示",选择"编辑 DataBinds"命令,单击 ImageUrl



图 5-27　以大图标显示商品信息模板

属性，在绑定代码表达式中输入 Eval("Ware_Image")。将 Image 控件显示图片路径动态绑定到 Ware_Image 数据字段。分别拖动 3 个 Label 到相应位置，同上操作，将 Text 属性分别绑定到 Ware_Number、Ware_Name、Ware_Price。拖动 ImageButton 控件到商品价格下方的单元格，单击 ImageUrl 属性，选择 WebIcon 文件夹中的 Buy. png 图像文件。

④ 使用 PagedDataSource 实现分页效果（见图 5-28）。拖动 Label 控件到 DataList 控件下方，更改 ID 为 lblCurrent。拖动 4 个 LinkButton 按钮到 lblcurrent 标签旁。设置 Text 属性分别为"首页"、"上一页"、"下一页"和"末页"，设置其 CommandName 分别为 First、Prev、Next、Last。

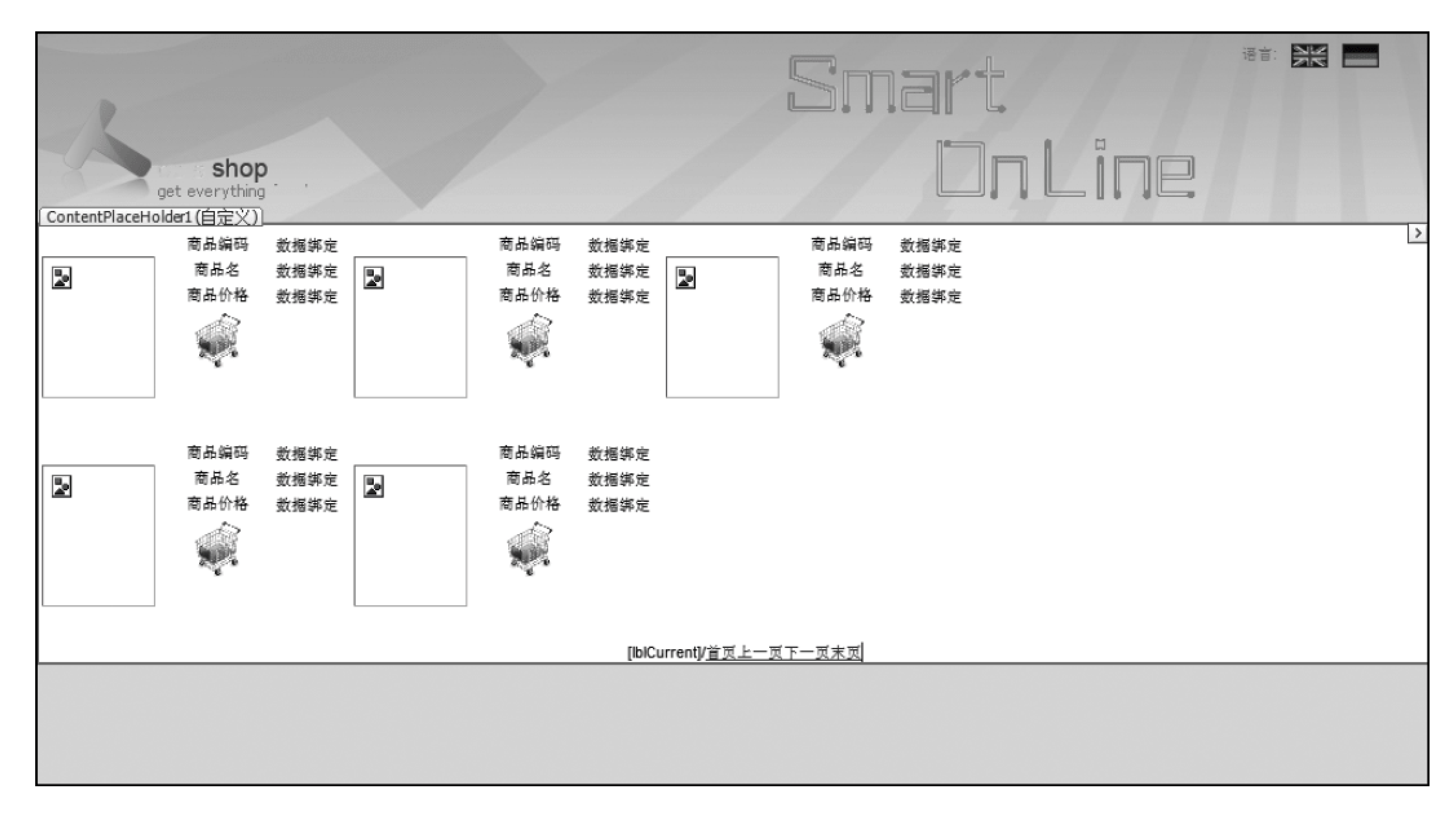


图 5-28　分页

按 F7 功能键切换到代码页面，编写 getData 方法。该方法要求提供页号这个参数，代码如下所示。使用 PagedDataSource 对象要设置 AllowPaging、PageSize 和 CurrentPageIndex 属性。

```
private void getData(int pageIndex)
{
    PagedDataSource pd= new PagedDataSource();
    pd.AllowPaging= true;
    pd.PageSize= 6;
    pd.CurrentPageIndex= pageIndex - 1;
    pd.DataSource= getData_Wares().DefaultView;
    DataList1.DataSource= pd;
    DataList1.DataBind();
    lblCurrent.Text= Convert.ToString(pd.CurrentPageIndex+ 1);
    lblCount.Text= "/";
    lblCount.Text+ = pd.PageCount.ToString();
    i= pd.PageCount;
    if(pd.IsFirstPage)
```

```
        {
            lbtnFirst.Visible=false;
            lbtnPrev.Visible=false;
        }
        else
        {
            lbtnFirst.Visible=true;
            lbtnPrev.Visible=true;
        }
        if(pd.IsLastPage)
        {
            lbtnLast.Visible=false;
            lbtnNext.Visible=false;
        }
        else
        {
            lbtnLast.Visible=true;
            lbtnNext.Visible=true;
        }
    }
```

5.4.2　任务 2：列表方式显示商品信息

1. 任务目标

（1）能熟练使用 GridView 显示数据。

（2）能熟练运用 ADO. NET 访问数据。

2. 任务内容

（1）使用 GridView 显示商品信息。

（2）以分页形式显示商品信息。

3. 任务实施步骤

该任务主要考察 GridView 控件的模板列、图像列和绑定列的使用和 ADO. NET 数据访问技术的再次使用。任务 2 的最终效果如图 5-29 所示。从图 5-29 中不难发现，左边第 1 列是采用图片列显示图片，第 2 列是采用模板列制作而成。

① 从 Layout. master 母版页创建内容页。鉴于这部分操作已经重复若干次，这里不再细述如何创建内容页。

② 拖动 GridView 控件到页面，单击“智能提示”，选择“编辑列”菜单项，打开“编辑列”对话框。单击 ImageField 项，单击“添加”按钮，单击 DataImageUrlField 属性，输入 Ware_Image。为控制图片显示的大小，需要设置控件的尺寸。单击 ControlStyle 的“＋”符号，设置 Height 属性值为 100px、Width 属性值为 80px。

③ 单击可用字段中的 TemplateField。单击“添加”按钮，增加模板列，单击“确定”按钮。单击 GridView 的“智能提示”，单击“编辑模板”菜单项，打开“项模板”窗口。

④ 拖动 HTML 标签中的 Table 控件到“项模板”窗口中，将其设置成 4 行 2 列，拖动

Label 控件到页面,如图 5-30 所示。单击 Label1 标签,单击"编辑 DataBinds",单击 Text 属性,在绑定代码表达式文本框中输入 Eval("Ware_Number")。同此操作,设置 Label2 的绑定表达式为 Eval("Ware_Name"),Label3 的绑定表达式为 Eval("Ware_Price",{0:C})。这里的{0:C}起格式化作用,将 Ware_Price 数据格式化成货币格式。拖动 ImageButton 到 Label3 下方单元格中,单击 ImageButton 按钮,在属性窗口中单击 ImageUrl 属性,选择 Buy.png 图像文件。



图 5-29　任务 2 效果图

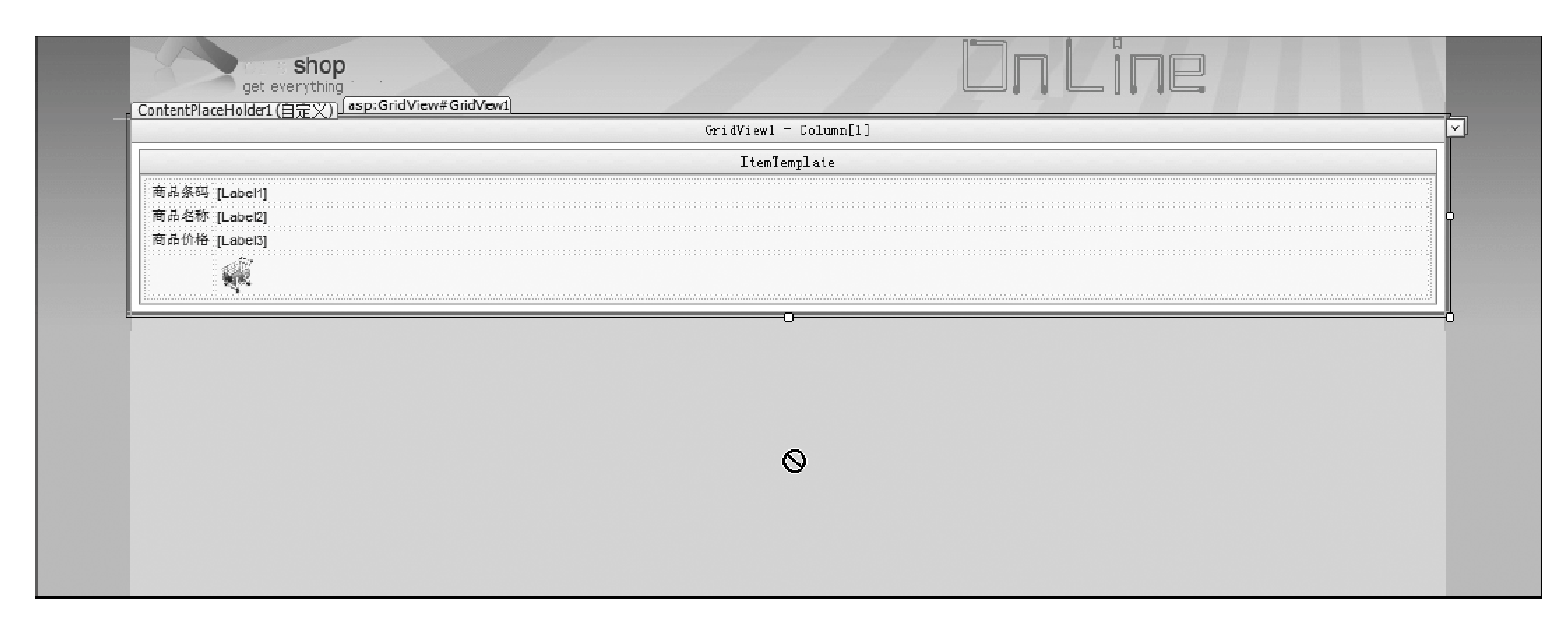


图 5-30　任务 2 模板效果图

⑤ 编写数据检索方法。同任务 1 类似。代码如下。

```
private DataTable getData_Wares()
{
    string sql="select * from T_Ware";
    SqlConnection conn;
```

```
        SqlCommand cmd;
        conn= sh.getConn();
        conn.Open();
        cmd= new SqlCommand(sql, conn);
        SqlDataAdapter sda= new SqlDataAdapter(cmd);
        DataTable dt= new DataTable();
        sda.Fill(dt);
        conn.Close();
        return dt;
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        if(!IsPostBack)
        {
            GridView1.DataSource= getData_Wares();
            GridView1.DataBind();
        }
    }
```

⑥ 设置分页功能。单击 GridView 控件,单击 AllowPaging 属性,将其设置为 True。单击事件按钮,双击 PageIndexChanging 事件,进入代码文件。输入下面的代码。这段代码需要设置 GridView 的页序号为事件参数 e 传递的 NewPageIndex,并重新绑定数据。

```
GridView1.PageIndex= e.NewPageIndex;
GridView1.DataSource= getData_Wares();
GridView1.DataBind();
```

5.4.3 任务 3:显示商品详细信息

1. 任务目标

(1) 能熟练操作 FormView 控件。

(2) 能通过 Url 传值。

2. 任务内容

(1) 使用 FormView 显示商品详细信息。

(2) 能使用 ADO. NET 技术获取数据。

3. 任务实施步骤

任务 3 要求当用户单击“详细”超链接时,显示该商品对应的详细信息(图 5-31)。

① 在任务 2 的模板中添加超链接。

② 从 Layout. master 母版页中生成内容页 WareDetails. aspx。拖动 FormView 到 WareDetails. aspx 页面。单击 FormView 控件的智能提示,单击“编辑模板”菜单项。

③ 拖动 HTML 标签中的 Table 控件到模板中,设置为 10 行 2 列。拖动 Image 控件到第 1 行第 2 列。单击“编辑 DataBinds”,单击 ImageUrl 属性,在绑定代码表达式中输入 Eval("Ware_Image")。

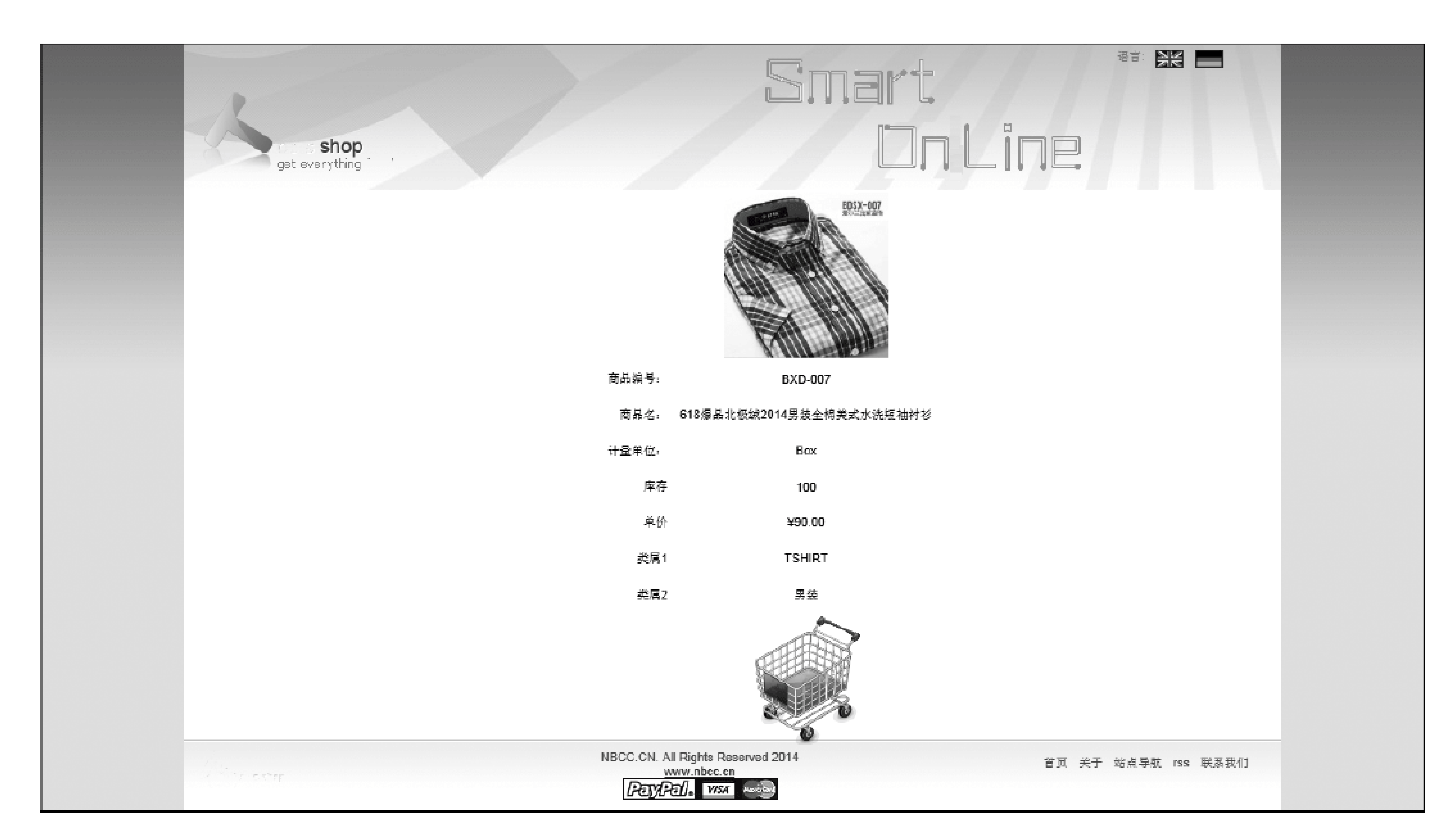


图 5-31　任务 3 效果图

④ 拖动 Label 到商品编码行，单击“编辑 DataBinds”，单击 Text 属性，在绑定代码表达式中输入 Eval("Ware_Number")。拖动 Label 到商品名行，单击“编辑 DataBinds”，单击 Text 属性，在绑定代码表达式中输入 Eval("Ware_Name")。拖动 Label 到计量单位行，单击“编辑 DataBinds”，单击 Text 属性，在绑定代码表达式中输入 Eval("Ware_Weight")。拖动 Label 到库存行，单击“编辑 DataBinds”，单击 Text 属性，在绑定代码表达式中输入 Eval("Ware_Stock")。

拖动 Label 到单价行，单击“编辑 DataBinds”，单击 Text 属性，在绑定代码表达式中输入 Eval("Ware_Price")。拖动 Label 到类属 1 行，单击“编辑 DataBinds”，单击 Text 属性，在绑定代码表达式中输入 Eval("Level3_Name")。拖动 Label 到类属 2 行，单击“编辑 DataBinds”，单击 Text 属性，在绑定代码表达式中输入 Eval("Level2_Name")。拖动 ImageButton 控件到第 9 行第 2 列，设置 ImageUrl 属性为 Buy.png。

⑤ 登录 SQL Server 2008，在查询窗口中创建视图。输入下列代码，创建视图 v_waredetails。

```
create view v_waredetails
as
SELECT      dbo.T_Ware.Ware_ID, dbo.T_Ware.Ware_Number,
dbo.T_Ware.Ware_Weight, dbo.T_Ware.Ware_Name,
dbo.T_Ware.Ware_stock, dbo.T_Ware.Ware_Price,
dbo.T_Ware.Ware_Image, dbo.T_Level3.Level3_Name, dbo.T_Level2.Level2_Name
FROM            dbo.T_Level3 INNER    JOIN
dbo.T_Level2 ON dbo.T_Level3.Level2_ID=dbo.T_Level2.Level2_ID INNER JOIN
dbo.T_Ware ON dbo.T_Level3.Level3_ID=dbo.T_Ware.Ware_Level3
```

⑥ 按 F7 功能键切换到代码页面，输入下列代码。getData 方法是从数据库检索商品信

息，将返回的商品信息存放到 DataTable 表中。当页面装载的时候，即在 Page_Load 事件中将 FormView 控件绑定到数据。

```
private DataTable getData()
{
    string sql="select * from v_waredetails";
    SqlConnection conn=new SqlConnection("Data Source=.;
        Initial Catalog=Smart;Integrated Security=True;
        MultipleActiveResultSets=true");
    SqlCommand cmd;
    conn.Open();
    cmd=new SqlCommand(sql, conn);
    SqlDataAdapter sda=new SqlDataAdapter(cmd);
    DataTable dt=new DataTable();
    sda.Fill(dt);
    conn.Close();
    return dt;
}
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {
        FormView1.DataSource=getData();
        FormView1.DataBind();
    }
}
```

5.5 总结归纳

子项目 5 的主要任务是以多种显示方式显示商品信息，主要运用了 DataList 和 GridView 控件进行数据显示。在这个实现过程中多次运用到 ADO. NET 技术访问数据库。后续内容中会编写 SqlHelper 通用类以简化 ADO. NET 的重复编写。DataList 必须是以模板的形式进行编辑，动态绑定的属性常以 Eval 方法进行动态绑定。GridView 提供了丰富的 7 种不同类型的列，便于开发者开发多用途的列，也通过模板列形式给开发者提供了更加灵活的、易于控制的方式控制 GridView。

5.6 课后习题

选择题

1. 单向数据绑定使用的方法是(　　)。

A. Eval　　B. Bind　　C. Bound　　D. DataBound

2. 双向数据绑定使用的方法是(　　)。

A. Eval　　B. Bind　　C. Bound　　D. DataBound

3. 数据绑定表达式包含在(　　)之内。

A. ＜!--和--＞　　B. /＊和＊/　　C. ＜％和％＞　　D. ＜％＃和％＞

4. 数据绑定控件要取到数据库表中更新前的旧值,应将该字段设置在(　　)属性中。

A. DataKeys　　B. PrimaryKeys

C. DataKeyNames　　D. DataKeyName

5. 数据绑定控件更新前的旧值的格式应设置在(　　)属性中。

A. OldValuesParameterFormat　　B. OldValueParameterFormat

C. OldValuesParameterFormatString　　D. OldValueParameterFormatString

6. 数据绑定控件的数据源参数不能绑定的参数源是(　　)。

A. QueryString　　B. Application　　C. Session　　D. Cookie

7. 数据绑定控件的数据源参数不能绑定的参数源是(　　)。

A. Application　　B. Form　　C. Session　　D. Control

8. 将 DropDownList 控件放入 GridView 中,应使用(　　)技术。

A. 母版页　　B. 模板列　　C. 动态列　　D. 选择项

9. GridView 中绑定了 DateTime 类型的字段,显示格式应在(　　)属性中设置。

A. DataFormatString　　B. DateFormatString

C. DataFormat　　D. DateFormat

10. GridView 中绑定了 DateTime 类型的字段,要使显示格式起作用,应设置行为(　　)。

A. HtmlEncodeFormatString＝True　　B. HtmlEncodeFormatString＝False

C. HtmlEncode＝True　　D. HtmlEncode＝False

11. GridView 中绑定一行并触发一次的事件是(　　)。

A. DataBound　　B. RowDataBound

C. DataBind　　D. RowDataBind

12. GridView 中数据全部绑定完成后触发的事件是(　　)。

A. DataBound　　B. RowDataBound

C. DataBind　　D. RowDataBind

13. DataList 控件中,增加各项之间的分隔符,应将＜hr /＞标签设置在(　　)模板中。

A. ItemTemplate　　B. AlternatingItemTemplate

C. SeparatorTemplate　　D. FooterTemplate

14. DataList 控件 dl 的页脚模板中有控件 tb,查找该对象的引用,以下正确的是(　　)。

A. dl.FooterRow.FindControl("tb");

B. dl.FooterRow.Cells[0].FindControl("tb");

C. dl.Rows[dl.Rows.Count －1].FindControl("tb");

D. dl.Controls[dl.Controls.Count －1].FindControl("tb");

5.7　同 步 操 练

在项目 4 中已经完成格林酒店管理系统用户注册和登录功能。项目经理要求开发人员增加客户管理模块，包括客户添加，如图 5-32 所示、客户编辑，如图 5-33 所示。单击“编辑”链接，返回到更新状态，允许用户输入，如图 5-34 所示。单击“删除”链接，出现“确认删除吗?”提示信息，如图 5-35 所示，单击“是”按钮，则删除该条信息。



图 5-32　添加客户

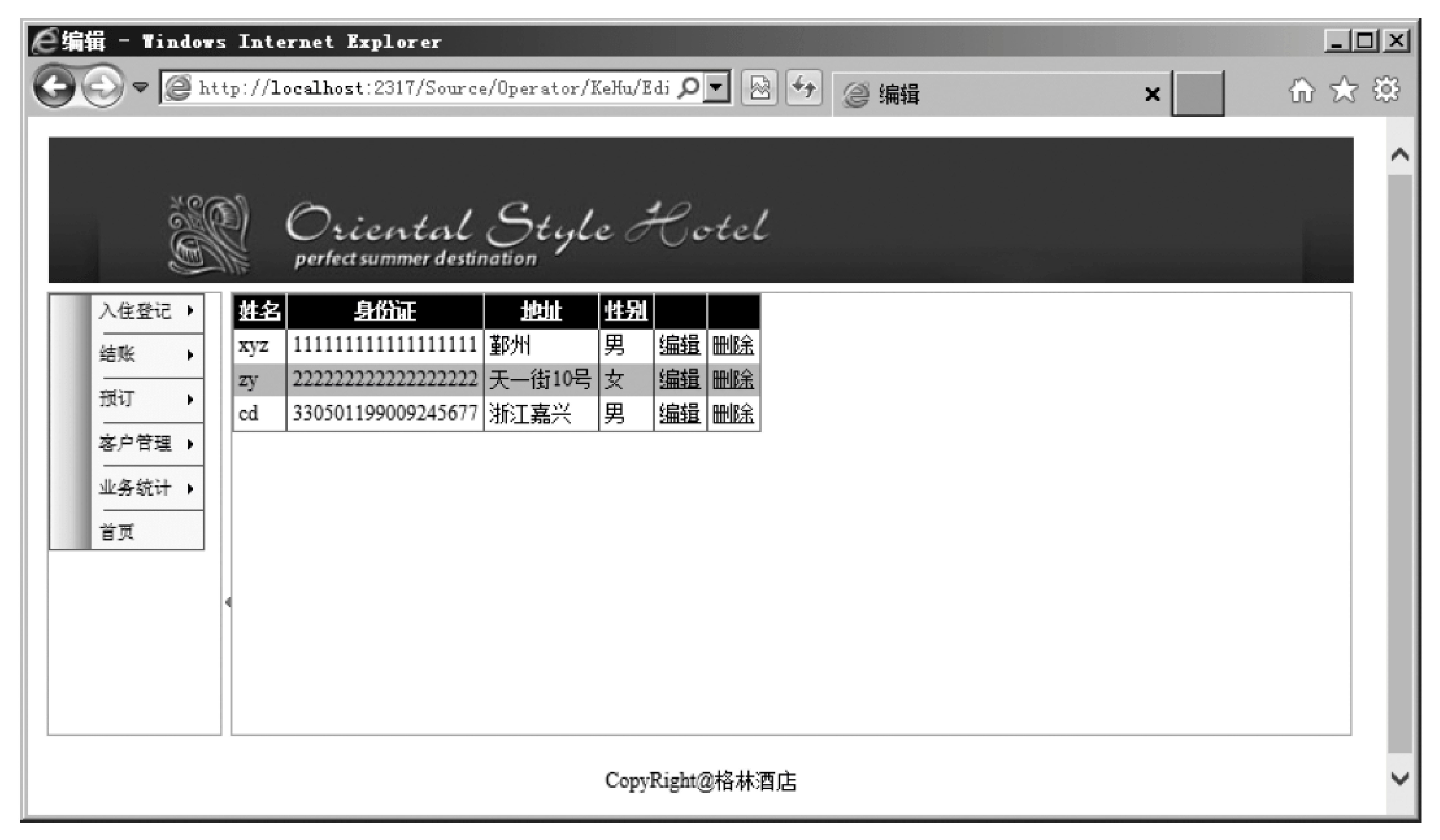


图 5-33　编辑客户信息



图 5-34　更新数据界面



图 5-35　删除客户

项目 6　会 员 购 物

6.1　项 目 引 入

会员购物是电子商城必须提供的一项功能。购物功能中的一个重要概念就是购物车。购物车是为消费者在网上购物中提供一个临时存储商品的地方。其主要功能包括：添加商品、删除商品、更改商品数量、商品金额小计、商品金额总计和清空购物车；还包括生成订单、打印订单、预览订单、提交订单和取消购物等。

6.2　项 目 分 析

经过小组讨论和分析，大家认为 Smart On Line 电子商城购物的操作流程如下：首先，登录到网站中浏览商品；其次，购买指定的商品，进入购物车页面中，在该页面可以实现更改商品数量、删除商品、清空购物车、继续购物等操作；最后，填写收货人信息，生成订单，并完成打印、预览、提交订单等操作。生成订单时必须登录系统以获取会员信息等。购物功能点分布比较分散，在两种商品显示页面之间存在购物功能链接。基于这种考虑，需要编写公共访问数据库类 SqlHelper，通过该类可以减少重复编写代码的工作量。进一步考虑数据库执行效率、页面代码和数据库之间的耦合，决定采用存储过程的形式访问数据库。在订单提交过程中需要同时在多表中进行操作，需要采用事务来保证操作的完整性。

6.3　知 识 准 备

6.3.1　存储过程

在 SQL Server 2008 中，存储过程和触发器是两个重要的数据库对象。使用存储过程，可以将 Transact-SQL(T-SQL)语句和控制流语句预编译到集合中并保存到服务器端，它使得管理数据库、显示关于数据库及其用户信息的工作变得更为容易。T-SQL 语句是应用程序与 SQL Server 数据库之间的主要编程接口，设计时大量的时间将花费在 T-SQL 语句和应用程序代码的编写上。在很多情况下，许多代码被重复使用多次，每次都输入相同的代码不但烦琐，而且在客户机上的大量命令语句逐条向 SQL Server 发送将降低系统运行效率，因此，SQL Server 提供了一种方法，它将一些固定的操作集中起来由 SQL Server 数据库服务器来完成，应用程序只需调用它的名称，即可实现某个特定的任务，这种方法就是存储过程。

存储过程是一种数据库对象，存储在数据库内，可由应用程序通过一个调用执行，而且允许用户声明变量并有条件地执行程序，具有很强的编程功能。存储过程可以使用EXECUTE语句来运行。在SQL Server中使用存储过程而不使用存储在客户端计算机本地的T-SQL程序，有以下几个方面的优点。

- 加快系统运行速度。存储程序只在创建时进行编译，以后每次执行存储过程都不需再重新编译，而一般SQL语句每执行一次就编译一次，所以使用存储过程可提高数据库执行速度。
- 封装复杂操作。当对数据库进行复杂操作时(如对多个表进行更新、删除时)，可用存储过程将此复杂操作封装起来与数据库提供的事务处理结合在一起使用。
- 实现代码重用，并可以实现模块化程序设计。存储过程一旦创建，以后即可在程序中调用任意多次，这可以改进应用程序的可维护性，并允许应用程序统一访问数据库。
- 增强安全性。可设定特定用户具有对指定存储过程的执行权限，而不具备直接对存储过程中引用的对象具有执行的权限。可以强制应用程序具有相应的安全性，参数化存储过程有助于使应用程序不受SQL注入式攻击。
- 减少网络流量。因为存储过程存储在服务器上，并在服务器上运行。一个需要数百行T-SQL代码的操作可以通过一条执行过程代码的语句来执行，而不需要在网络中发送数百行代码，这样就可以减少网络流量。

1. 存储过程分类

存储过程是一个被命名的存储在服务器上的T-SQL语句的集合，是封装重复性工作的一种方法，它支持用户声明的变量、有条件执行的程序和其他强大的编程功能。

在SQL Server 2008中，存储过程可以分为三类：系统存储过程、用户存储过程和扩展性存储过程。

(1) 系统存储过程

系统存储过程是由SQL Server系统提供的存储过程，可以作为命令来执行各种操作。系统存储过程主要用来从系统表中获取信息，为系统管理员管理SQL Server提供帮助，为用户查看数据库对象提供方便。例如，执行SP_HELPTEXT系统存储过程可以显示规则、默认值、未加密的存储过程、用户函数、触发器或视图的文本信息；执行sp_depends系统存储过程可以显示有关数据库对象相关性的信息；执行sp_rename系统存储过程可以更改当前数据库中用户创建对象的名称。SQL Server中许多管理工作是通过执行系统存储过程来完成的，许多系统信息也可以通过执行系统存储过程而获得。系统存储过程定义在系统数据库master中，其前缀是sp_。在调用时不必在存储过程前加上数据库名。

(2) 用户存储过程

用户存储过程是指用户根据自身需要，为完成某一特定功能，在用户数据库中创建的存储过程。用户创建存储过程时，存储过程名的前面加上“##”表示创建全局临时存储过程；在存储过程名前面加上“#”，表示创建局部临时存储过程。局部临时存储过程只能在创建它的会话中可用，当前会话结束时则删除它。全局临时存储过程可以在所有会话中使用，即所有用户均可以访问该过程。它们都保存在tempdb数据库上。

存储过程可以接受输入参数、向客户端返回表格或者标量结果和消息、调用数据定义语

言(DDL)和数据操作语言(DML),然后返回输出参数。在 SQL Server 2008 中,用户定义的存储过程有两种类型:T-SQL 或 CLR,如表 6-1 所示。

表 6-1　用户定义存储过程的两种类型

存储过程类型	说　　明
T-SQL	T-SQL 存储过程是指保存的 T-SQL 语句集合,可以接受和返回用户提供的参数。存储过程也可能从数据库向客户端应用程序返回数据
CLR	CLR 存储过程是指对 Microsoft .NET Framework 公共语言运行时方法的引用,可以接受和返回用户提供的参数。它们在.NET Framework 程序中是作为类的公共静态方法实现的

(3) 扩展性存储过程

扩展性存储过程通过在 SQL Server 环境外执行的动态链接库(DLL,Dynamic-Link Libraries)来实现。扩展存储过程通过前缀“xp_”来标识,它们用与存储过程相似的方式来执行。

2. 使用存储过程

在使用存储过程之前,首先需要创建一个存储过程,这可以通过 T-SQL 语句 CREATE PROCEDURE 来完成。在使用的过程中,包括对存储过程的执行、查看和修改以及删除操作。

(1) 创建存储过程

在 SQL Server 2008 中,可以使用 T-SQL 语句 CREATE PROCEDURE 来创建存储过程。在创建存储过程时,应该指定所有的输入参数、执行数据库操作的编程语句、返回至调用过程或批处理时以示成功或失败的状态值、捕获和处理潜在错误时的错误处理语句等。在设计和创建存储过程时,应该满足一定的约束和规则,只有满足了这些约束和规则才能创建有效的存储过程。在 CREATE PROCEDURE 定义中不能出现的语句如表 6-2 所示。

表 6-2　CREATE PROCEDURE 定义中不能出现的语句

CREATE AGGREGATE	CREATE RULE
CREATE DEFAULT	CREATE SCHEMA
CREATE(或 ALTER)FUNCTION	CREATE(或 ALTER)TRIGGER
CREATE(或 ALTER)PROCEDURE	CREATE.(或 ALTER)VIEW
SET PARSEONLY	SET SHOWPLAN_ALL
SET SHOWPLAN_TEXT	SET SHOWPLAN_XML
USE Database_name	

注意:

- 可以引用在同一存储过程中创建的对象,只要引用时已经创建了该对象即可。
- 可以在存储过程内引用临时表。
- 如果在存储过程内创建本地临时表,则临时表仅为该存储过程而存在;退出该存储过程后,临时表将消失。
- 如果执行的存储过程将调用另一个存储过程,则被调用的存储过程可以访问由第一个存储过程创建的所有对象,包括临时表在内。

- 如果执行对远程 SQL Server 2008 实例进行更改的远程存储过程，则不能回滚这些更改，而且远程存储过程不参与事务处理。
- 存储过程中的参数的最大数目为 2100。
- 存储过程中的局部变量的最大数目仅受可用内存的限制。
- 根据可用内存的不同，存储过程最大可达 128MB。

(2) 创建存储过程的语法

使用 CREATE PROCEDURE 语句创建存储过程的语法如下。

```
CREATE PROCDURE procedure_name[;number]
[{@parameter data_type}
[VARYING][=default][OUTPUT]][,…n]
[WITH
{RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION}]
[FOR REPLICATION]
AS sql_statement[…n]
```

其主要参数含义如下。

- Procedure_name：新存储过程的名称。过程名称在架构中必须唯一，可在 Procedure_name 前面使用一个数字符号"#"来创建局部临时过程，使用两个数字符号"#"来创建全局临时过程。对于 CLR 存储过程，不能指定临时名称。
- ;number 是可选的整数，用来对同名的过程分组。使用一个 DROP PROCEDURE 语句可将这些分组过程一起删除。如果名称中包含分隔标识符(;)，则数字不应该包含在标识符中，另外，只应在 procedure_name 前使用分隔符。
- @parameter：过程中的参数。在 CREATE PROCEDURE 语句中可以声明一个或多个参数。除非定义了参数的默认值或者将参数设置为等于另一个参数，否则用户必须在调用过程时为每个声明的参数提供值，如果指定了 FOR REPLICATION，则无法声明参数。
- Data_type：参数的数据类型。所有数据类型均可以用作存储过程的参数。不过 cursor 数据类型只能用于 OUTPUT 参数。如果指定的数据类型为 cursor，则还必须指定 VARYING 和 OUTPUT 关键字。对于 CLR 存储过程，不能指定 char、varchar、text、next、image、cursor 和 table 作为参数。如果参数的数据类型为 CLR 用户定义类型，则必须对此类型有 EXECUTE 权限。
- Default 参数的默认值。如果定义了 dafault 值，则无须指定此参数的值即可执行过程。默认值必须是常量或 NULL。如果过程使用带 like 关键字的参数，则可包含下列通配符：%、_、[]、[^]。
- Output：指示这是输出参数。此选项的值可以返回给调用 EXECUTE 的语句。使用 OUTPUT 参数将值返回给过程的调用方。除非是 CLR 过程，否则 text、ntext 和 image 参数不能用作 OUTPUT 参数。OUTPUT 关键字的输出参数可以为游标占位符，CLR 过程除外。＜sql_statement＞要包含在过程中的一个或多个 T-SQL 语句中。

(3) 使用图形工具创建存储过程

除了直接编写 T-SQL 语句创建存储过程以外，SQL Server 2008 还提供了一种简便的

方法，即使用 SQL Server Management Studio 工具。操作步骤如下。

① 打开 SQL Server Management Studio 窗口，连接到 BookDateBase 数据库。

② 依次展开“服务器”|“数据库”|BookDateBase|“可编程性”节点。

③ 从列表中右击“存储过程”节点，选择“新建存储过程”命令，然后将出现如图 6-1 所示的显示 CREATE PROCEDURE 语句的模板，可以修改要创建的存储过程的名称，然后加入存储过程所包含的 SQL 语句。

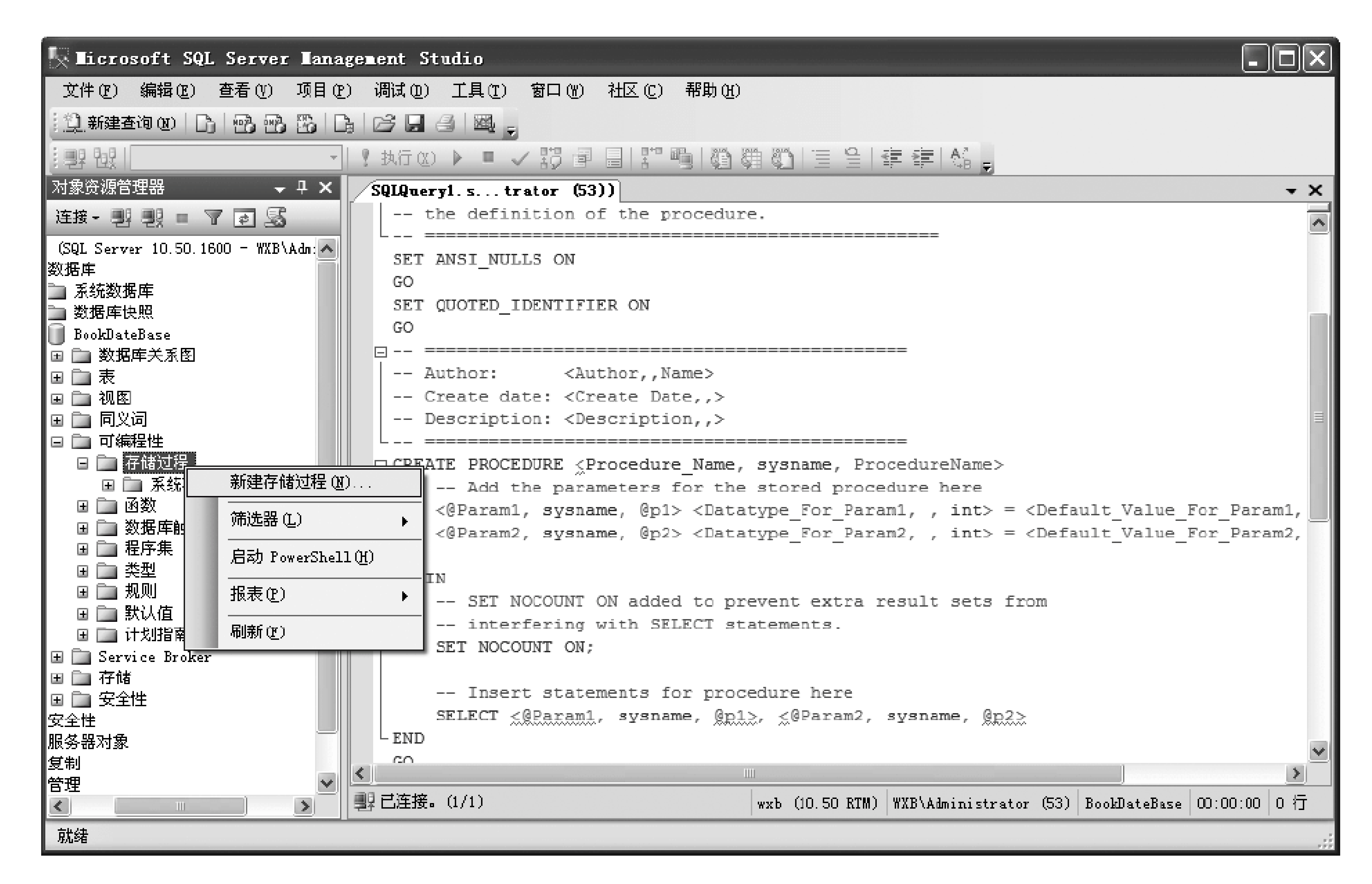


图 6-1　创建存储过程

④ 修改完后，单击“执行”按钮，即可创建一个存储过程。

示例 1：在 SQL Server 2008 的示例数据库 BookDatebase 中创建一个名为 Reader_proc 的存储过程，它将从表中返回所有读者的姓名、性别、电话、等级。使用 CREATE PROCEDURE 语句如下。

```
Use BookDatebase
Go
CREATE PROCEDURE Reader_proc
As
SELECT Rname,Rsex,Rphone,rleve
FROM Reader
```

示例 2：创建存储过程 proc_GetCountsBook，获取 BookDatebase 数据库中图书的总数量。具体语句如下。

```
Use BookDatebase
Go
CREATE PROCEDURE proc_GetCountsBook
As
SELECT count(ID) AS 总数 FROM Books
```

以上两个存储过程示例都是从单个表中提取数据。在示例 2 中使用了简单的表达式。

示例 3：下面使用 SELECT 语句链接多个表，最终返回借书人的简明信息。存储过程名称是 proc_BorRreader，创建语句如下。

```
Use BookDatebase
Go
CREATE PROCEDURE proc_BorR_reader
As
SELECT B.Bnum,B.Bname,B.writer,R.Rcert,R.Rname,BR.botime
From Books B,Reader R, BorrowORreturn BR
WHERE B.Bnum=BR.Bnum and R.Rcert=BR.Rcert and BR.botime<>''
```

（4）执行存储过程

在需要执行存储过程时，可以使用 T-SQL 语句中的 EXECUTE 命令。如果存储过程是批处理中的第一条语句，那么不使用 EXECUTE 关键字也可以执行该存储过程。EXECUTE 语法格式如下。

```
[ { EXEC | EXECUTE } ]
{
[ @return_status= ]
{ procedure_name [;number] | @procedure_name_var }
@parameter= [ { value | @variable [ OUTPUT ] | [ DEFAULT ] } ]
[,...n]
[ WITH RECOMPILE ]
```

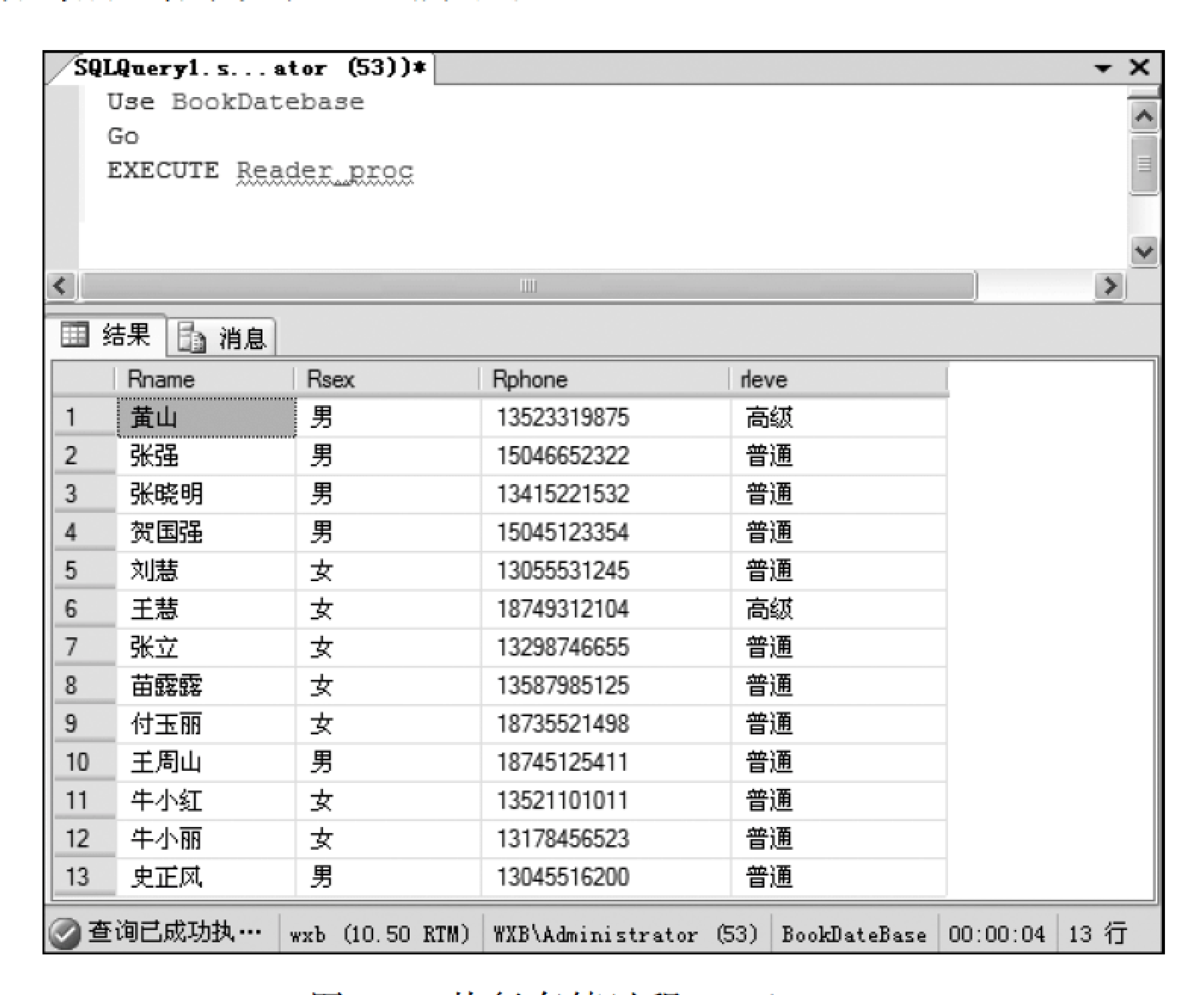
其中主要参数的含义如下。

- @return_status：这是一个可选的整型变量，保存存储过程的返回状态。这个变量在用于 EXECUTE 语句前，必须在批处理、存储过程或函数中声明过。
- Procedure_name：要调用的存储过程名称。
- ;number：这是可选的整数，用于将相同名称的过程进行组合，使得它们可以用 DROP PROCEDURE 语句删除。在 BookDatebase 数据库中使用的过程可以用“Reader_proc;1”、proc_GetCountsBook 等来命名。DROP PROCEDURE Reader_proc 语句将除去整个组。在对过程分组后，不能删除组中的单个过程。例如，语句“DROP PROCEDURE proc_GetCountsBook;2”是不允许的。
- @procedure_name_var：局部定义变量名，代表存储过程名称。
- @parameter：过程参数，在 CREATE PROCEDURE 语句中定义。参数名称前必须加上符号“@”。
- Value：过程中参数的值。如果参数名称没有指定，参数值必须以 CREATE PROCEDURE 语句中定义的顺序给出。
- @variable：用来保存参数或者返回参数的变量。
- OUTPUT：指定存储过程必须返回一个参数。该存储过程的匹配参数也必须由关键字 OUTPUT 创建。使用游标变量作参数时使用该关键字。
- DEFAULT：根据过程的定义，提供参数的默认值。当过程需要的参数值是没有事先定义好的默认值，或缺少参数，或指定了 DEFAULT 关键字，就会出错。

示例 4：通过 EXECUTE 语句来依次执行示例 1、2、3 创建的 3 个存储过程。首先是执行 Reader_proc 存储过程，它位于 BookDatebase 数据库中，语句如下：

```
Use BookDatebase
Go
EXECUTE Reader_proc
```

执行上述语句后，结果如图 6-2 所示。



	Rname	Rsex	Rphone	rleve
1	黄山	男	13523319875	高级
2	张强	男	15046652322	普通
3	张晓明	男	13415221532	普通
4	贺国强	男	15045123354	普通
5	刘慧	女	13055531245	普通
6	王慧	女	18749312104	高级
7	张立	女	13298746655	普通
8	苗露露	女	13587985125	普通
9	付玉丽	女	18735521498	普通
10	王周山	男	18745125411	普通
11	牛小红	女	13521101011	普通
12	牛小丽	女	13178456523	普通
13	史正风	男	13045516200	普通

图 6-2　执行存储过程 Reader_proc

然后再使用同样的方法，执行 BookDatebase 数据库中的其他两个存储过程，结果分别如图 6-3 和图 6-4 所示。



	总数
1	14

图 6-3　执行存储过程 proc_GetCountsBook

除使用 EXECUTE 直接执行存储过程之外，还可以将存储过程嵌入 INSERT 语句中执行。这样操作时，INSERT 语句将把本地或远程存储过程返回的结果集加入一个本地表中。SQL Server 2008 会将存储过程中的 SELECT 语句返回的数据载入表中，其前提是该表必须存在并且其数据类型必须匹配。



图 6-4　执行存储过程 proc_BorR_reader

3. 存储过程的参数

存储过程的优势不仅在于存储在服务器端、运行速度快，还有重要的一点就是存储过程可完成的功能非常强大，特别是在 SQL Server 2008 中。本节将学习如何在存储过程使用参数，包括输入参数和输出参数，以及参数的默认值等。

存储过程可以使用两种类型的参数：输入参数和输出参数。参数用于在存储过程以及应用程序之间交换数据，其中：

- 输入参数允许用户将数据值传递到存储过程或函数中。
- 输出参数允许存储过程将数据值或游标变量传递给用户。
- 每个存储过程向用户返回一个整数代码，如果存储过程没有显式设置返回代码的值，则返回代码为 0。

存储过程的参数在创建时应在 CREATE PROCEDURE 和 AS 关键字之间定义，每个参数都要指定参数名和数据类型，参数名必须以@符号作为前缀，可以为参数指定默认值；如果是输出参数，则应用 OUTPUT 关键字描述。各个参数定义之间要用逗号隔开，具体语法如下：

```
@parameter_name data_type [=default ] [ OUTPUT ]
```

（1）输入参数

输入参数，即指在存储过程中有一个条件，在执行存储过程时为这个条件指定值，再通过存储过程返回相应的信息。使用输入参数可以向同一存储过程多次查找数据库。例如，可以创建一个存储过程用于返回 BookDatebase 数据库上某条借阅信息中包括的图书名称。通过为同一存储过程指定不同的借阅者，来返回不同的图书名称。

在示例 3 中创建的存储过程 proc_BorR_reader 只能对表进行特定的查询。若要使这个存储过程更加通用化、灵活且能够查询某个类别中相应的图书信息，那么读者信息中的读者卡号就应该是可变的，这样的存储过程才能返回某个类别的图书信息。在这个存储过程上将一个读者的卡号作为参数来实现，名称为 proc_GetReaderBooks，代码如下。

```
USE [BookDateBase]
GO
CREATE PROCEDURE [dbo].[proc_GetReaderBooks]
@Rcert int
As
SELECT B.Bnum,B.Bname,B.writer,R.Rname,BR.botime,R.Rcert
From BooksB,ReaderR,BorrowORreturnBR
WHERE B.Bnum=BR.BnumANDR.Rcert=BR.RcertANDBR.botime<>''ANDBR.Rcert=@Rcert
```

以上代码创建了一个名为 proc_GetReaderBooks 的存储过程，使用一个字符串型的参数@Rcert 来执行。执行带有输入参数的存储过程时，SQL Server 2008 提供了如下两种传递参数的方式。按位置传递这种方式是在执行存储过程的语句中直接给出参数的值。当有多个参数时，给出的参数的顺序与创建存储过程的语句中的参数顺序一致，即参数传递的顺序就是参数定义的顺序。使用这种方式执行 proc_GetReaderBooks 存储过程的代码为：

```
EXEC proc_GetReaderBooks '10010'
```

这种方式是在执行存储过程的语句中，使用“参数名＝参数值”的形式给出参数值。通过参数名传递参数的好处是，参数可以按任意顺序给出。用这种方式执行 proc_GetReaderBooks 存储过程的代码如下，执行结果如图 6-5 所示。

```
EXEC proc_GetReaderBooks @Rcert='10010'
```

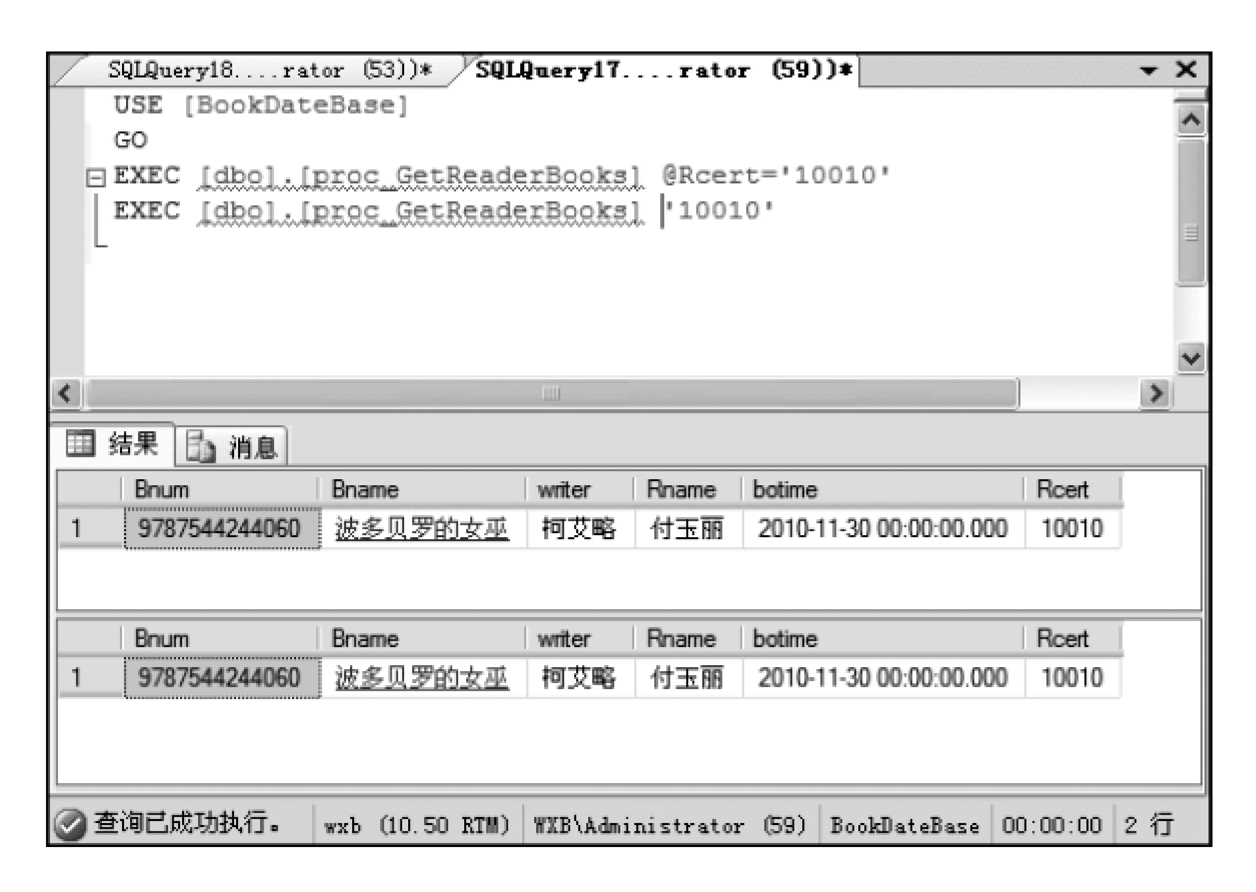


图 6-5　存储过程的执行结果

(2) 使用默认参数值

执行存储过程 proc_GetReaderBooks 时，如果没有指定参数，则系统运行时就会出错；如果希望不给出参数时也能够正确运行，则可以通过给参数设置默认值来实现。因此，如果要将 proc_GetReaderBooks 存储过程修改为默认值，使用类别编号为 10010 的 proc_GetReaderBooks，则可以运行下列代码。

```
USE [BookDateBase]
GO
CREATE PROCEDURE [dbo].[proc_GetReaderBooks]
@Rcert int=10010
As
SELECT B.Bnum,B.Bname,B.writer,R.Rname,BR.botime,R.Rcert
```

```
From BooksB,ReaderR,BorrowORreturnBR
WHERE B.Bnum=BR.BnumANDR.Rcert=BR.RcertANDBR.botime<>''ANDBR.Rcert=@Rcert
```

(3) 输出参数

通过定义输出参数,可以从存储过程中返回一个或多个值。为了使用输出参数,必须在 CREATE PROCEDURE 语句和 EXECUTE 语句中指定关键字 OUTPUT。在执行存储过程时,如果忽略 OUTPUT 关键字,存储过程仍会执行,但没有返回值。

```
USE [BookDateBase]
GO
CREATE PROCEDURE [dbo].[proc_GetReaderBookscount]
@Rcert int=10010
@bookcounts int OUTPUT
As
SELECT @bookcount=COUNT(B.Bnum)
From Books B,Reader R, BorrowORreturn BR
WHERE B.Bnum=BR.BnumANDR.Rcert=BR.RcertANDBR.botime<>''ANDBR.Rcert=@Rcert
```

以上代码创建一个名为 proc_GetReaderBooks1 的存储过程,它使用两个参数:@Rcert 为输入参数,用于指定要查询的读者编号,默认参数值为 10010;@bookcounts 为输出参数,用来返回读者借阅的图书数量。

为了接收某一存储过程的返回值,需要一个变量来存放返回参数的值,在该存储过程的调用语句中,必须为这个变量加上 OUTPUT 关键字来声明。下面的代码显示了如何调用 proc_GetReaderBooks1,并将得到的结果返回到@bookcounts 中,其运行结果如图 6-6 所示。

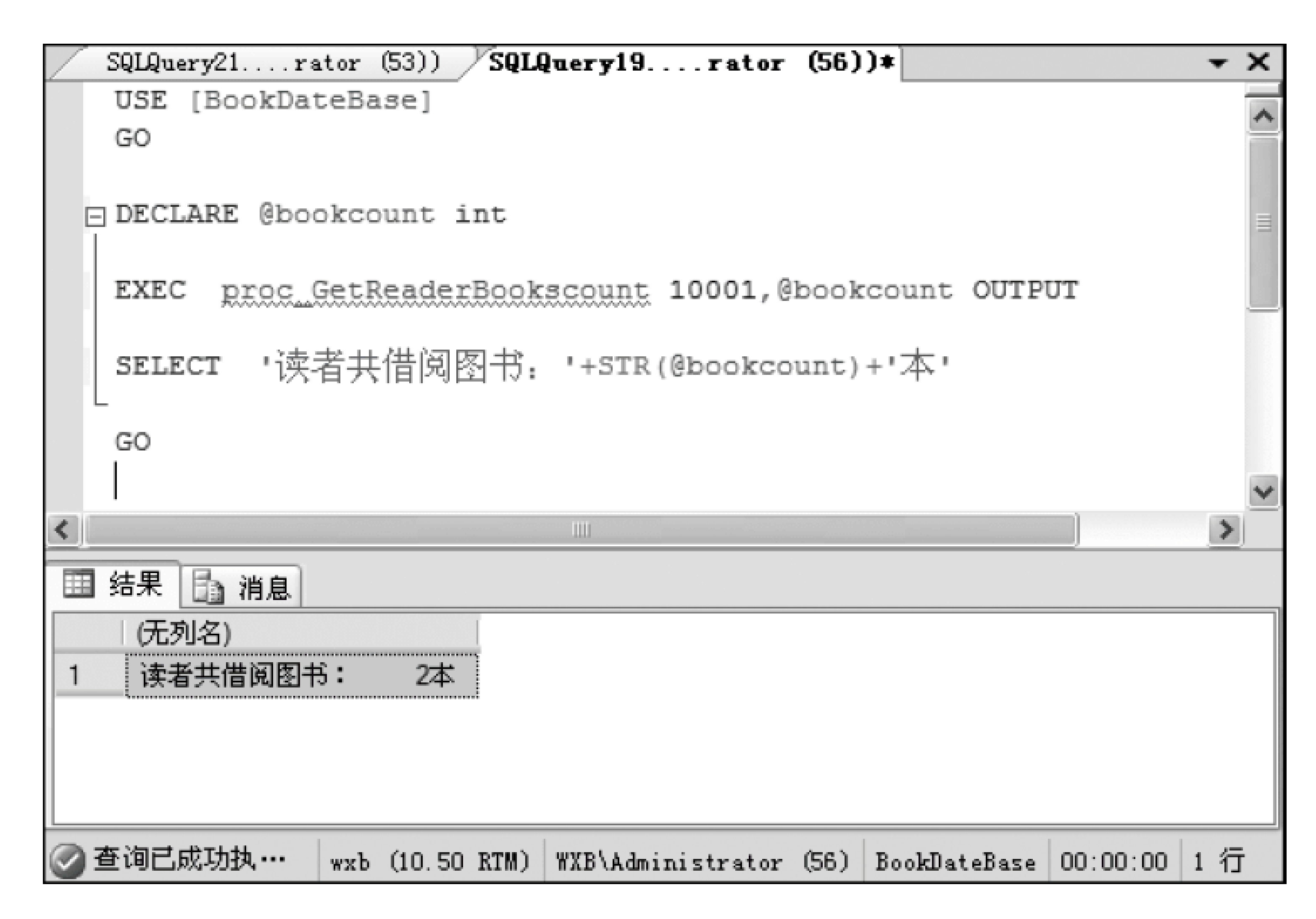


图 6-6 带输出参数的存储过程

```
USE [BookDateBase]
GO
DECLARE    @bookcount int
EXEC  proc_GetReaderBookscount 10001,@bookcount OUTPUT
SELECT '读者共借阅图书:'+STR(@bookcount)+'本'
GO
```

（4）存储过程的返回值

存储过程在执行后都会返回一个整型值。如果执行成功，则返回 0；否则返回 −1～−99 的随机数，也可以使用 RETURN 语句来指定一个存储过程的返回值。

示例 5：下面创建一个名为 aAndb 的存储过程，用于计算出两个参数的和。本例使用 SET 语句，但是也可以使用 SELECT 语句来组织一个字符串，语句如下。

```
CREATE PROC aANDb
@a int=0,@b int=0,@c int=0 OUTPUT
AS
Set @c=@a+@b
Return @c
```

@c 参数由 OUTPUT 关键字指定。在执行这个存储过程时，需要指定一个变量存放返回值，然后再显示出来。如下所示为一个调用这个存储过程的示例：

```
DECLARE @int c int
EXEC aANDb 6,2,@int c OUTPUT
SELECT '两个之和为：'+STR(@INT C)
```

执行如果如图 6-7 所示。

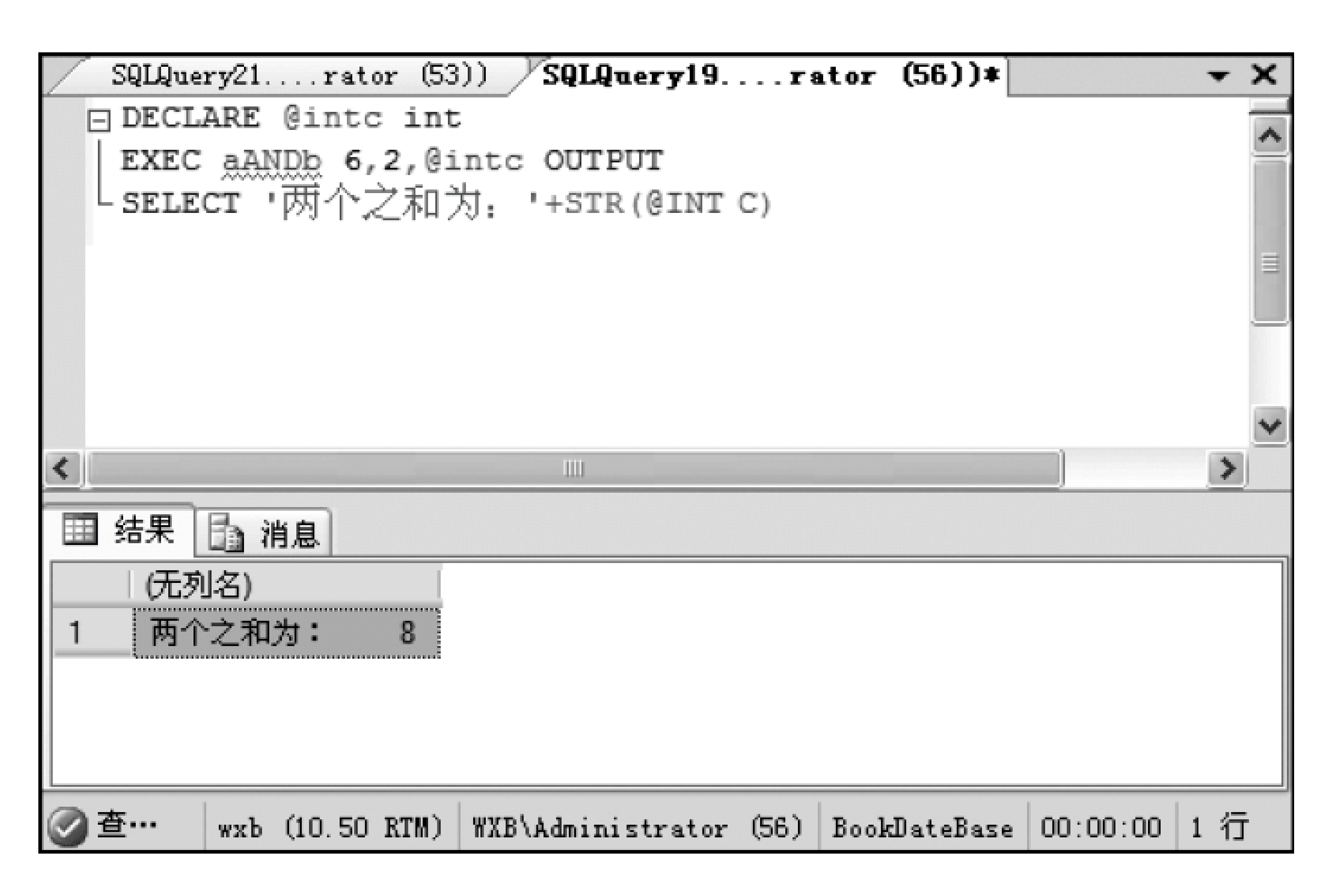


图 6-7　执行 aANDb 存储过程的结果

4. 删除存储过程

使用 DROP PROCEDURE 语句可以从当前的数据库中删除用户定义的存储过程。删除存储过程的基本语法如下。

```
DROP PROCEDURE {procedure}
```

下面的语句将删除 sp_delete 存储过程：

```
DROP PROC sp_delete
```

如果另一个存储过程调用某个已被删除的存储过程，SQL Server 2008 将在执行调用进程时显示一条错误消息。但是，如果定义了具有相同名称和参数的新存储过程来替换已被删除的存储过程，那么引用该过程的其他过程仍能成功执行。

5. 修改存储过程

使用 ALTER PROCEDURE 语句来修改现有的存储过程与删除和重建存储过程不同，因为它仍保持存储过程的权限不发生变化。在使用 ALTER PROCEDURE 语句修改存储过程时，SQL Server 2008 会覆盖以前定义的存储过程。修改存储过程的基本语句如下。

```
ALTER PROCEDURE procedure_name[;number]
[{@parameter data_type}
[VARYING][=default][OUTPUT]]
[,…n]
[WITH
{ RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION}]
[FOR REPLICATION]
AS
sql_statement[…n]
```

修改存储过程的语法中的各个参数与创建存储过程语法中的各个参数相同，这里不再重复介绍。在使用 ALTER PROCEDURE 语句时，应考虑以下方面的事项：

- 如果要修改具有任何选项的存储过程，例如 WITH ENCRYPTION 选项，必须在 ALTER PROCEDURE 语句中包括该选项以保留该选项提供的功能。
- ALTER PROCEDURE 语句只能修改一个单一的过程，如果过程调用了其他存储过程，嵌套的存储过程不受影响。
- 在默认状态下，允许该语句的执行者是存储过程最初的创建者、sysadmin 服务器角色成员和 db_owner 与 db_ddladmin 固定的数据库角色成员，用户不能授权执行 ALTER PROCEDURE 语句。

6.3.2 事务

事务是一组组合成逻辑工作单元的数据库操作，在系统执行过程中可能会出错，但事务将控制与维护每个数据库的一致性和完整性。事务处理的主要特征是，任务要么全部完成，要么都不完成。在写入一些记录时，要么写入所有记录，要么什么都不写入。如果在写入一个记录时出现了一个失败，那么在事务处理中已写入的其他数据就会回滚。事务可能由很多单个任务构成。

简单事务的一个常见例子：把钱从 A 账户转到 B 账户，这涉及两项任务，即从 A 账户把钱取出来，再把钱存入 B 账户。两项任务要么同时成功，要么一起失败后给予回滚，以便保持账户的状态和原来相同。否则，在执行某一个操作的时候可能会因为停电、网络中断等原因而出现故障，所以有可能更新了一个表中的行，但没有更新相关表中的行。如果数据库支持事务，则可以将数据库操作组成一个事务，以防止因这些事件而使数据库出现不一致。

事务的 ACID 属性如下。

- 原子性(Atomicity)：事务的所有操作是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。原子性消除了系统处理操作子集的可能性。
- 一致性(Consistency)：数据从一种正确状态转换到另一种正确状态。事务在完成时，必须使所有的数据都保持一致。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。当事务结束时，所有的内部数据结构都必须是

正确的。在存款取款的例子中，逻辑规则是，钱是不能凭空产生或销毁的，对于每个(收支)条目必须有一个相应的支出条目产生，以保证账户是动态收支平衡的。

- 隔离性(Isolation)：由并发事务所作的修改必须与任何其他并发事务所作的修改隔离。查看数据所处的状态，要么是事务修改它之前的状态，要么是事务修改它之后的状态。简单的理解就是，防止多个并发更新彼此干扰。事务在操作数据时与其他事务操作隔离。隔离性一般是通过加锁的机制来实现的。
- 持久性(Durability)：事务完成之后，它对于系统的影响是永久性的。已提交的更改即使在发生故障时也依然存在。

对于事务的开发，.NET 平台提供了 3 种非常简单方便的事务机制：ADO. NET 级别的事务，ASP. NET 页面级别的事务和系统(System)级别的事务。

1. ADO. NET 级别的事务

现在我们对事务的概念和原理都有所了解了，并且作为已经有一些基础的 C# 开发者，我们已经熟知以下编写数据库交互程序的一些要点。

(1) 使用 SqlConnection 类的对象的 Open()方法建立与数据库服务器的连接。

(2) 将该连接赋值给 SqlCommand 对象的 Connection 属性。

(3) 将欲执行的 SQL 语句赋值给 SqlCommand 的 CommandText 属性。

(4) 通过 SqlCommand 对象进行数据库操作。

创建一个 ADO. NET 事务是很简单的，需要定义一个 SqlTransaction 类型的对象。SqlConnection 和 OleDbConnection 对象都有一个 BeginTransaction 方法，它可以返回 SqlTransaction 或者 OleDbTransaction 对象。然后赋给 SqlCommand 对象的 Transcation 属性，即实现了二者的关联。为了使事务处理可以成功完成，必须调用 SqlTransaction 对象的 Commit()方法；如果有错误，则必须调用 Rollback()方法撤销所有的操作。基于以上认识，下面就开始动手写一个基于 ADO. NET 的事务处理程序。

```
string conString="data source=127.0.0.1;database=codematic;user id=sa;
password=";
SqlConnection myConnection=new SqlConnection(conString);
myConnection.Open();
//启动一个事务
SqlTransaction myTrans=myConnection.BeginTransaction();
//为事务创建一个命令
SqlCommand myCommand=new SqlCommand();
myCommand.Connection=myConnection;
myCommand.Transaction=myTrans;
try
{
    myCommand.CommandText="update P_Product set Name='笔记本 1' where Id=52";
    myCommand.ExecuteNonQuery();
    myCommand.CommandText="update P_Product set Name='电脑 3' where Id=53";
    myCommand.ExecuteNonQuery();
    myTrans.Commit();            //提交
    Response.Write("两条数据更新成功");
}
catch(Exception ex)
```

```
{
    myTrans.Rollback();            //遇到错误,回滚
    Response.Write(ex.ToString());
}
finally
{
    myConnection.Close();
}
```

ADO.NET 事务的优势在于简单,事务可以跨越多个数据库进行访问,独立于数据库,不同数据库的专有代码都被隐藏了。但也存在一些限制,事务执行在数据库连接层上,所以需要在执行事务的过程中手动地维护一个连接。

2. ASP.NET 页面级别的事务

ASP.NET 事务可以说是在.NET 平台上事务实现方式最简单的一种,仅仅需要一行代码即可。在 aspx 的页面声明中加一个额外的属性,即事务属性 Transaction="Required",它有如下值:Disabled(默认)、NotSupported、Supported、Required 和 RequiresNew,这些设置和"COM+"及企业级服务中的设置一样。典型的一个例子是如果你想在页面上下文中运行事务,那么要将其设置为 Required。如果页面中包含了用户控件,那么这些控件也会包含到事务中,事务会存在于页面的每个地方。需要在执行事务的页面中先声明事务:

```
<%@ Page Transaction="Required"  Language="C#" AutoEventWireup="true"
    CodeBehind="WebForm3.aspx.cs" Inherits="WebApplication4.WebForm3" %>
```

然后在页面代码中使用 using System.EnterpriseServices 导入 ContextUtil 所在类的命名空间。最后调用 ContextUtil 类的 SetComplete 或者 SetAbort 方法进行事务的提交或撤销。例如下面代码。

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        Work1();
        Work2();
        ContextUtil.SetComplete();     //提交事务
    }
    catch(System.Exception except)
    {
        ContextUtil.SetAbort();          //撤销事务
        Response.Write(except.Message);
    }
}
private void Work1()
{
    string conString="data source=127.0.0.1;database=codematic;user id=sa;
        password=";
    SqlConnection myConnection=new SqlConnection(conString);
    string strSql="Insert Into P_Category(CategoryId,Name)values('1',
        'test1')";
```

```
        SqlCommand myCommand=new SqlCommand(strSql, myConnection);
        myConnection.Open();
        int rows=myCommand.ExecuteNonQuery();
        myConnection.Close();
    }
    private void Work2()
    {
        string conString="data source=127.0.0.1;database=codematic;user id=sa;
            password=";
        SqlConnection myConnection=new SqlConnection(conString);
        string strSql="Insert Into P_Category(CategoryId,Name)values('2',
            'test2')";
        SqlCommand myCommand=new SqlCommand(strSql, myConnection);
        myConnection.Open();
        int rows=myCommand.ExecuteNonQuery();
        myConnection.Close();
    }
```

ContextUtil 是用于获取“COM＋”上下文信息的首选类。由于此类的成员全部为 static，因此在使用其成员之前不需要对此类进行实例化。ASP.NET 页面事务的优势是实现简单，不需要额外的编码。但是页面的所有代码都是同一个事务，这样的事务可能会很大，而需要的通常是分散的、小的事务。

3. 系统级别的事务

.NET Framework 2.0 以后增加了系统级别的事务 System.Transactions，这是一种新的命名空间，完全专注于控制事务性行为。引入了执行事务性工作的更简单方法及一些新的性能优化。

System.Transactions 提供了一个“轻量级”的、易于使用的事务框架。通过 System.Transactions，则只要简单的几行代码，用户也根本不需要考虑是简单事务还是分布式事务，系统会自动根据事务中涉及的对象资源判断使用何种事务管理器。简而言之，对于任何的事务，用户只要使用同一种方法进行处理即可。下面介绍 System.Transactions 的常见用法。

首先要引用：

```
using System.Transactions;
```

其次，将事务操作代码放在 TransactionScope 中执行。如：

```
using(TransactionScope ts=new TransactionScope())
{
    //事务操作代码
    ts.Complete();
}
```

这是最简单、也是最常见的用法。创建了新的 TransactionScope 对象后，即开始创建事务范围。如代码示例所示，建议使用 using 语句创建范围。位于 using 块内的所有操作将成为一个事务的一部分，因为它们共享其所定义的事务执行上下文。本例中的最后一行，调用 TransactionScope 对象的 Complete 方法，将导致退出该块时请求提交该事务。此方法还提

供了内置的错误处理，出现异常时会终止事务。例如下面的代码段所示。

```
using(TransactionScope ts=new TransactionScope())                    //使整个代码块成为事务性代码
{
    #region 在这里编写需要实现事务的代码
    string msg="";
    string conString="data source=127.0.0.1;database=codematic;
        user id=sa; password=";
    SqlConnection myConnection=new SqlConnection(conString);
    myConnection.Open();
    SqlCommand myCommand=new SqlCommand();
    myCommand.Connection=myConnection;
    try
    {
        myCommand.CommandText="update P_Product set Name='电脑 2'
            where Id=52";
        myCommand.ExecuteNonQuery();
        myCommand.CommandText="update P_Product set Name='电脑 3'
            where Id=53";
        myCommand.ExecuteNonQuery();
        msg="成功!";
    }
    catch(Exception ex)
    {
        msg="失败:"+ex.Message;
    }
    finally
    {
        myConnection.Close();
    }
    #endregion
    ts.Complete();
    return msg;
}
```

上面的代码演示了在一个 TransactionScope 对象里面打开一个数据库连接的过程。这个数据库连接由于处在一个 TransactionScope 对象里面，所以会自动获得事务处理的能力。如果这里数据库连接的是 SQL Server 2008，那么这个事务将不会激活一个 MSDTC 管理的分布式事务，而是会由.NET 创建一个局部事务，性能非常高。再看下面的例子。

```
void MethodMoreConn()
{
    using(TransactionScope ts=new TransactionScope())
    {
        using(SqlConnection conn=new SqlConnection(conString1))
        {
            conn.Open();
            using(SqlConnection conn2=new SqlConnection(conString2))
            {
                conn2.Open();
```

```
            }
        }
        ts.Complete();
    }
}
```

这个例子更加充分地说明了 TransactionScope 对象功能的强大，使两个数据库完成了连接。虽然上面的 conn 和 conn2 是两个不同的连接对象，可能分别连接到不同的数据库，但是由于它们处在一个 TransactionScope 对象中，它们就具备了“联动”的事务处理能力。在这里，将自动激活一个 MSDTC 管理的分布式事务。

虽然.NET 对事务提供了很好的支持，但是没有必要总是使用事务。使用事务的第一条规则是，在能够使用事务的时候都应该使用事务，但是不要过度使用。原因在于，每次使用事务都会占用一定的内存开销。另外，事务可能会锁定一些表的行。还有一条使用事务的规则是，只有当操作需要的时候才使用事务。例如，如果只是从数据库中查询一些记录，或者执行单个查询，则在大部分时候都不需要使用显式事务。

开发人员应该在头脑中始终保持一个概念，就是用于修改多个不同表数据的冗长事务会严重妨碍系统中的所有其他用户。这很可能导致一些性能问题。当实现一个事务时，遵循下面的实践经验能够达到可接受的结果。

- 避免使用在事务中的 Select 语句返回数据，除非语句依赖于返回数据。
- 如果使用 Select 语句，则只选择需要的行，这样不会锁定过多的资源，而尽可能地提高性能。
- 尽量将事务全部写在 T-SQL 或者 API 中。
- 避免事务与多重独立的批处理工作结合，应该将这些批处理放置在单独的事务中。
- 尽可能避免大量更新。

另外，必须注意的一点就是事务的默认行为。在默认情况下，如果没有显式地提交事务，则事务会回滚。虽然默认行为允许事务的回滚，但是显式回滚方法总是一个良好的编程习惯。这不仅仅只是释放锁定的数据，也将使得代码更容易读取并且会更少出现错误。.NET 提供的事务功能很强大，具体的内容远不止本书所讲解的这样简单。本书只是起到一个抛砖引玉的功能。希望读者能够灵活恰当地使用事务功能，而不要过度使用事务，否则可能会对系统的性能起到负面的作用。

6.3.3　DataTable

DataTable 是一个保存在内存中的数据表，可以通过拖放工具栏里面的控件来创建和使用它，也可以在编写程序过程中根据需要独立创建和使用，最常见的情况是作为 DataSet 的成员使用，在这种情况下就需要用在编程过程中并根据需要动态创建数据表。本小节主要讲述用编码的方式来建立 DataTable 数据表以及进行相应的操作。

1. 用代码创建 DataTable 数据表

通过添加对象的方式可以直接在 DataSet 中创建数据表，可以通过使用 Add 方法将 DataTable 添加到 DataSet 中。那么在程序中怎么创建 DataTable 数据表呢？在程序中创建 DataTable 对象可以使用 DataTable 构造函数创建一个表名为 TableName 的数据表，代码如下：

```
DataTable NewTable=new DataTable(TableName);
```

另外也可以通过以下方法创建DataTable对象：使用DataAdapter对象的Fill方法或FillSchema方法在DataSet中创建，使用这些方法的前提是在与数据库相连接的情况下。特别值得注意的是，将一个DataTable作为成员添加到一个DataSet的Tables集合中后，不能再将其添加到任何其他DataSet的表集合中。使用DataTable构造函数初次创建DataTable时，是没有架构的(即结构中没有列)。没有架构的DataTable数据是没有办法使用的，因此在使用这种DataTable数据表之前要定义表的架构，必须创建DataColumn对象并将其添加到表的Columns集合中。创建DataTable时，不需要为TableName属性提供值，可以在其他时间指定该属性，或者将其保留为空，这些都不影响DataTable的使用。应该注意的是，在将一个没有TableName值的表添加到DataSet中时，该表会得到一个从Table(表示Table0)开始递增的默认名称TableN。下面的代码表示创建了DataTable对象的实例，并为其指定名称Customers。

```
DataTable workTable=new DataTable("Customers");
```

以下代码是将创建的DataTable实例，即Customers表添加到DataSet的Tables集合中。实现代码如下：

```
DataSet customers=new DataSet();
DataTable customersTable=customers.Tables.Add("CustomersTable");
```

2. 用编程方式添加DataTable列

前面已经学过如何使用代码来创建DataTable。要定义表的架构，除了可以创建DataColumn对象并将其添加到表的Columns集合中之外，也可以为表定义主键列，可以创建Constraint约束对象并将其添加到表的Constraints约束集合中。DataColumn类型表示了DataTable上的一列。总的来说，绑定到某个DataTable的所有DataColumn类型的集合就表示一个表。DataTable包含了由表的Columns属性引用的DataColumn对象的集合。这个列的集合与任何约束一起来定义表的架构(即结构)。通过使用DataColumn构造函数，或者通过调用表的Columns属性的Add方法，可在表内创建DataColumn对象。Add方法将接受可选的ColumnName、DataType参数，并将创建新的DataColumn作为集合的成员。它还会接受现有的DataColumn对象并将其添加到集合中，也会根据请求返回对所添加的DataColumn的引用。

示例6：向DataTable中添加了4列，代码如下，CustID、CustLName、CustFName、Purchases分别是Customers中的列名。

```
DataTable workTable=new DataTable("Customers");
DataColumn workCol=workTable.Columns.Add("CustID");
workTable.Columns.Add("CustLName");
workTable.Columns.Add("CustFName");
workTable.Columns.Add("Purchases");
```

3. 设置DataTable数据表的主键

数据库开发的一个通常规则就是表至少要有一个列作为主键。主键约束用于唯一标识给定表中的一条记录(行)。

示例 7：假设现在需要新建一个 DataColumn 列来表示 EmpID 字段并且要将这个列作为表的主键，它必须有 AllowDBNull 和 Unique 属性，实现代码如下。

```
DataTable workTable= new DataTable("Customers");
DataColumn workCol= workTable.Columns.Add("CustID", typeof(Int32));
workCol.AllowDBNull= false;
workCol.Unique= true;
workTable.Columns.Add("CustLName", typeof(String));
workTable.Columns.Add("CustFName", typeof(String)));
workTable.Columns.Add(" urchases", typeof(String)));
```

示例 8：将 CustID 列的属性设置为不允许为 DBNull 值，并将值约束为唯一。但是，如果将 CustID 列定义为表的主键列，AllowDBNull 属性就会自动设置为 false，并且 Unique 属性会自动设置为 true。

4. 设置列的数据类型

通过上面的学习，大家应该已经知道怎么向新建的数据表中添加列了，那么下面来看一看，怎么为添加的列设置数据类型。数据类型是标明一列数据的数据类型属性，根据不同的需要，可以在 DataTable 数据表中建立不同的列，并可以为不同的列设置不同的数据类型。继续以示例 7 中的 Customers 表为例，现为其创建的列添加数据类型，代码如下。

```
//创建一个数据表 Customers
DataTable CustomersTable= new DataTable("Customers");
//创建一个 Int32类型、名称是 CustID的列,把这个列设置成主键,并且不允许为空
DataColumn CustomersCol= CustomersTable.Columns.Add("CustID", typeof(Int32));
CustomersCol.AllowDBNull= false;
CustomersCol.Unique= true;
//创建三个 String类型的列 CustLName、CustFName、Purchases
CustomersTable.Columns.Add("CustLName", typeof(String));
CustomersTable.Columns.Add("CustFName", typeof(String)));
CustomersTable.Columns.Add(" urchases", typeof(String)));
```

例中将 CustID 列定义为表的主键列。CustID 列指定的数据类型是 Int32，CustLName 列、CustFName 列、Purchases 列指定的数据类型都是 String 类型的列，当然也可以不设置列的数据类型，在这个时候 DataColumn 的 DataType 属性默认为字符串类型，当然可以根据需要在创建列名时进行列数据类型的设置。

5. 启用 Autoincrementing 字段

在设置完 DataColumn 列的数据类型以后，也可以像 SQL Server 数据库表一样，把某一列设置成自动递增的。简单地说，自动增加列的功能可以确保当一个新行被添加到给定表时，可以基于当前的递增步长值自动指定这个列的值。特别是当某一列作为没有重复值的主键的时候，这个功能就特别有用。在 DataTable 中，这个功能可以用 AutoIncrement（是否将列的值自动递增）、AutoIncrementSeed（起始值、种子值）和 AutoIncrementStep（步长）属性来控制。

示例 9：创建一个支持自动递增的 DataColumn 列。标记列的起始值是 0，步长值为 1，代码如下。

```
//创建一个新列
DataColumn myColumn= new DataColumn();
myColumn.ColumnName= " CustID ";
myColumn.DataType= System.Type.GetType("System.Int32");
//设置自动递增
myColumn.AutoIncrement= true;
myColumn.AutoIncrementSeed= 0;
myColumn.AutoIncrementStep= 1;
```

创建一个数据类型为 Int32 的 CustID 列，为了能使这个字段的值实现自动增加的效果，需要把列的 AutoIncrement 属性设置为 true；把列的种子值 AutoIncrementSeed 设为 0，也就是从 0 开始计数；同时设置自动增加的步长 AutoIncrementStep 为 1，每次增加一个值。由于种子值被设为 0，前面 5 个值应该是 0、1、2、3 和 4。

6. 用编程方式添加 DataTable 行

在为 DataTable 定义了架构之后，也就是设置好了需要的列名以后，就可以通过将 DataRow 对象添加到表的 Rows 集合中来将数据行添加到表中。与添加 DataColumn 类似，同样可以通过使用 DataRow 构造函数，或者通过调用表的 Rows 属性的 Add 方法，可在表内创建 DataRow 对象。DataColumn 对象集合表示了表的模式(Schema)。DataTable 通过内部的 DataColumnCollection 类型保存表中所有的列。相反，DataRow 类型集合就表示表中的实际数据。这样，如果 Customers 表中有 10 个记录，就可以使用 10 个 DataRow 类型来表示它们。使用 DataRow 类的成员可以对表中的值进行插入、删除、求值和操作操作。

示例 10：向 Customers 表中插入行，实现代码如下。

```
//创建一个 Customers 数据表
DataTable CustomersTable= new DataTable("Customers ");
//创建一个新的数据行
DataRow arow= CustomersTable.NewRow();
//设置行的值
arow[ColumnName]= DataValue;
//把数据行添加到已经创建的 Customers 数据表中
CustomersTable.Rows.Add(arow);
```

这段代码表示新建一行 arow，并给这行中的某一个列名赋值为 DataValue，最后把这一行添加到 Customers 表中。使用 DataRow 与使用 DataColumn 有些不同，因为不可以直接创建这个类型的实例，而是获得一个来自给定 DataTable 的引用。假设想向 Customers 表中添加新行，则可以用 DataTable. NewRow()方法获得下一空位，然后在上面填充每列的数据。

7. 用编程方式删除 DataTable 行

从 DataTable 对象中删除 DataRow 对象的方法有两种：DataRowCollection 对象的 Remove 方法和 DataRow 对象的 Delete 方法。Remove 方法和 Delete 方法都可以将 DataTable 的行 DataRow 删除，但是前者是从 DataRowCollection 中删除 DataRow，而后者只将行标记为“删除”，当应用程序调用 AcceptChanges 方法时，才会发生实际的删除。通过使用 Delete 方法，可以在实际删除之前先以编程方式检查哪些行标记为“删除”。如果将行

标记为“删除”，其 RowState 属性会设置为 Deleted。在将 DataSet 或 DataTable 与 DataAdapter 和关系型数据源一起使用时，用 DataRow 的 Delete 方法移除行。Delete 方法只是在 DataSet 或 DataTable 中将行标记为 Deleted，而不会移除它。而 DataAdapter 在遇到标记为 Deleted 的行时，会执行其 DeleteCommand 方法，以便在数据源中删除该行，然后就可以用 AcceptChanges 方法永久移除该行。如果使用 Remove 删除该行，则该行将从表中完全移除，但 DataAdapter 不会在数据源中删除该行。

6.3.4 GridView 的高级应用技巧

GridView 对象中的 RowDataBound 事件为开发人员提供了方便的控制行、列数据的途径。要获取当前行的某个数据列，可以使用如下几种方法。

- Cells[x]. Txt：从列单元格的文本值获取。这种方法简单且高效，最为常用，但功能单纯。存在的缺点是：无法获取设置了隐藏属性的数据列的值，所取到的值为空。只能获取在 HTML 中定义过的数据列，无法查询数据源中的当前数据行的所有字段列。
- e. Row. Cells[x]. FindControl("YourcontrolName")：用于在单元格内查找某个服务器控件，从而获得其数据值。这种方式可以操作单元格内的服务器控件。一般用于处理模板列中的数据或控件。
- (DataRowView)e. Row. DataItem). Row. ItemArray[i]. ToString()：方法的核心是 e. Row. DataItem，它是 GridView 中 Object 类型的行数据集，将其转化为 DataRowView 类型后，可以获得更多的操作方法。此数据集表示数据源当前行的全部字段列，ItemArray[i]是当前行全部字段列的数组对象，可以通过索引 i 获得任意字段值。此方法的有利之处是可以对数据源的全部字段进行查询。

1. 在 GridView 中进行鼠标指针的移入、移出及变色

在创建 GridView 控件时，必须先为 GridView 的每一行创建一个 GridViewRow 对象。创建每一行时，将引发一个 RowCreated 事件；当行创建完毕，每一行 GridViewRow 就要绑定数据源中的数据，当绑定完成后，将引发 RowDataBound 事件。那么利用 RowDataBound 事件就可以控制每一行添加客户端的 js 代码。代码如下。

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if(e.Row.RowType==DataControlRowType.DataRow)
    {
        e.Row.Attributes.Add("onmouseover", "currentcolor=this.style.
        backgroundColor;this.style.backgroundColor='#e0ffff'");
        e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor=
        currentcolor");
    }
}
```

该段代码表示为每个行添加一个 onmouseover 属性，这个属性其实就是对应 JavaScript 中的“鼠标移入”事件，该事件触发时将当前的背景色保存到变量 currentcolor 中，接着设置当前行的颜色为＃e0ffff。同样也添加了“鼠标移出”事件，这个事件发生时行

的背景色设置为 currentcolor，即恢复到原先的背景色。

2. 在 GridView 中利用 ShowFooter 进行数据汇总

通常需要在 GridView 中对某一列数据进行汇总，并将汇总结果显示在脚注中，如图 6-8 所示。

ID	Product	Price	UnitsInStock
31	Gorgonzola Telino	￥12.50	0
32	Mascarpone Fabioli	￥32.00	9
33	Geitost	￥2.50	112
34	Sasquatch Ale	￥14.00	111
35	Steeleye Stout	￥18.00	20
36	Inlagd Sill	￥19.00	112
37	Gravad lax	￥26.00	11
38	Côte de Blaye	￥263.50	17
39	Chartreuse verte	￥18.00	69
40	Boston Crab Meat	￥18.40	123
Total value in stock (on this page): ￥12,880.70			
1 2 3 4 5 6 7 8			

footer

图 6-8 数据汇总

这个功能的实现依赖于 GridView 的 DataBound 事件。首先需要设置 GridView 的 ShowFooter 属性为 True，默认值为 False。然后在 GridView 的 DataBound 事件中输入如下代码。

```
protected void gridSummary_DataBound(object sender, EventArgs e)
{
    decimal valueInStock=0;
    //遍历 GridView 所有行中指定的列,这里遍历第 3 列和第 4 列
    foreach(GridViewRow row in gridSummary.Rows)
    {
        decimal price=Decimal.Parse(row.Cells[2].Text.Substring(1));
        int unitsInStock=Int32.Parse(row.Cells[3].Text);
        valueInStock+=price * unitsInStock;
    }
    GridViewRow footer=gridSummary.FooterRow;
    //设置 footer 跨越这个行,并设置为居中对齐
    footer.Cells[0].ColumnSpan=3;
    footer.Cells[0].HorizontalAlign=HorizontalAlign.Center;
    //移除不需要的单元格
    footer.Cells.RemoveAt(2);
    footer.Cells.RemoveAt(1);
    //组合结果,复制给 footer 的第一个单元格
    footer.Cells[0].Text="Total value in stock(on this page):"+
              valueInStock.ToString("C");
}
```

汇总行信息与网格的其他行拥有相同个数的列。所以如果希望跨越多个单元格显示文字，需要配置适当的单元格的 ColumSpan 属性以跨越多格。本例中，第一个单元格跨越了三列。

6.4　项 目 实 施

6.4.1　任务 1：编写 SqlHelper 数据访问类

1. 任务目标

（1）能使用 Visual Studio 2012 生成新的类。

（2）能熟练生成新方法。

2. 任务内容

（1）创建 SqlHelper 类。

（2）在 SqlHelper 类中添加访问数据库的方法。

3. 任务实施步骤

该任务需要掌握类和方法的概念。此外为安全起见，需要将连接字符串放置在 web.config 配置文件中以便项目发布时进行加密。

（1）新建 SqlHelper 类

右击 App_Code 文件夹，在弹出菜单中单击“添加”→“添加新项”命令，打开“添加新项”对话框。在“名称”文本框中输入 SqlHelper，单击“类”列表项，如图 6-9 所示，再单击“添加”按钮。



图 6-9　新建 SqlHelper 类

（2）双击打开 web.config 文件，在＜connectionStrings＞小节中添加连接字符串内容。代码如下。

```
<connectionStrings>
    <add name="SmartConnectionString"
        connectionString="Data Source=.;Initial Catalog=Smart;
        Integrated Security=True;
```

```
        MultipleActiveResultSets= true"
        providerName= "System.Data.SqlClient"/>
</connectionStrings>
```

<add>表示在<connectionStrings>小节中添加了一个名字为 SmartConnectionString 的连接字符串。DataSource 表示访问的服务器为本地服务器,Initial Catalog 表示访问的数据库为 Smart,Integrated Security 表示采用集成访问的形式访问数据库,MultipleActiveResultSets的作用是指定多活动的结果集是否与指定的链接相互关联。类型是 bool 类型,true 代表与指定的链接关联;false 代表与指定的链接不关联;默认是 false。

(3) 双击 SqlHelper. cs 类文件来打开它,输入下列代码。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.SqlClient;
using System.Data;
///< summary>
///SqlHelper 的摘要说明
///< /summary>
public class SqlHelper
{
SqlConnection con= new SqlConnection(System.Configuration.ConfigurationManager.
ConnectionStrings["SmartConnectionString"].ToString());
    SqlDataReader sdr;
    public SqlHelper()
    {
        //
        //TODO: 在此处添加构造函数的代码
        //
    }
    public SqlConnection getConn()
    {
        return con;
    }

    public SqlConnection returnConn()
    {
        return con;
    }

    public SqlDataReader QueryOperation(String StrQueryCommand)
    {
        if(con.State= = System.Data.ConnectionState.Closed)
            con.Open();
        SqlCommand cmd= new SqlCommand();
        cmd.Connection= con;
        cmd.CommandText= StrQueryCommand;
        if(sdr != null)
            sdr.Close();
```

```
        sdr=cmd.ExecuteReader();
        return sdr;
    }
    public bool ExeNonQuery(String StrCmd)
    {
        if(con.State==System.Data.ConnectionState.Closed)
            con.Open();
        SqlCommand cmd=new SqlCommand();
        cmd.Connection=con;
        cmd.CommandText=StrCmd;
        try
        {
            cmd.ExecuteNonQuery();

        }
        catch(Exception ex)
        {
            throw ex;
            return false;
        }
        return true;

    }
    public void closeConn()
    {
        if(con.State !=System.Data.ConnectionState.Closed)
            con.Close();
    }

    public bool ExeNoQueryProc(String cmdName, SqlParameter[] ps)
    {
        if(con.State==System.Data.ConnectionState.Closed)
            con.Open();
        SqlCommand cmd=new SqlCommand();
        cmd.Connection=con;
        cmd.CommandText=cmdName;
        cmd.CommandType=System.Data.CommandType.StoredProcedure;
        foreach(SqlParameter p in ps)
        {
            cmd.Parameters.Add(p);
        }
        try
        {
            cmd.ExecuteNonQuery();

        }
        catch(Exception ex)
        {

            return false;
        }
```

```
            return true;
        }
        public DataTable QueryOperationProc(String cmdName, SqlParameter[] ps)
        {
            if(con.State==System.Data.ConnectionState.Closed)
                con.Open();
            SqlCommand cmd=new SqlCommand();
            cmd.Connection=con;
            cmd.CommandText=cmdName;
            cmd.CommandType=System.Data.CommandType.StoredProcedure;
            if(ps !=null)
            {
                foreach(SqlParameter p in ps)
                {
                    cmd.Parameters.Add(p);
                }
            }
            DataTable dt=new DataTable();
            SqlDataAdapter sda=new SqlDataAdapter(cmd);
            sda.Fill(dt);
            return dt;
        }

        public DataTable QueryProc(String cmdStr, SqlParameter[] ps)
        {
            DataTable dt=new DataTable();
            if(con.State==System.Data.ConnectionState.Closed)
                con.Open();
            SqlCommand cmd=new SqlCommand();
            cmd.Connection=con;
            cmd.CommandType=CommandType.StoredProcedure;
            cmd.CommandText=cmdStr;
            if(ps !=null)
            {
                foreach(SqlParameter p in ps)
                {
                    cmd.Parameters.Add(p);
                }
            }
            SqlDataAdapter sda=new SqlDataAdapter(cmd);
            sda.Fill(dt);
            con.Close();
            return dt;
        }

    }
```

下面分别讲解这个类文件中的各个属性和方法。SqlConnection con = new SqlConnection 语句用于创建一个连接对象，SqlConnection 对象需要提供一个连接字符串参数。本文中已经将连接字符串配置在 web.config 配置文件中。所以需要通过以下语句

来获取配置文件中的 SmartConnectionString 的连接字符串内容。

```
System.Configuration.ConfigurationManager.ConnectionStrings["SmartConnection-
String"]
```

getConn()方法的作用是返回一个连接对象。QueryOperation 方法要求提供一个 SQL 查询操作的代码，返回的是一个存放了数据的 SqlDataReader 对象，需要注意的是，QueryOperation 方法只能提供查询功能，不能提供非查询功能。ExeNonQuery 方法提供非查询功能，需要调用者提供一个非查询的 SQL 代码，如果成功执行则返回 true，否则返回 false。closeConn 方法是将创建的连接关闭。QueryOperation 和 ExeNonQuery 方法都是需要直接传递一段 SQL 代码。QueryOperationProc 方法则是 ADO. NET 调用存储过程的实现方式。QueryOperationProc 方法要求调用者提供存储过程的名称和相应的存储过程参数。存储过程参数用 SqlParameter 数组形式进行传递。ADO. NET 调用存储过程的方法跟直接调用 SQL 代码有所不同，必须设置 SqlCommand 对象的 CommandType 属性为 StoredProcedure，即：

```
cmd.CommandType= System.Data.CommandType.StoredProcedure;
```

另外，必须将存储过程的参数添加到 SqlCommand 对象的 Parameters 集合中。

```
if(ps != null)
{
    foreach(SqlParameter p in ps)
    {
        cmd.Parameters.Add(p);
    }
}
```

QueryOperationProc 返回的是 DataTable 类型的一个对象。ExeNoQueryProc 方法也是需要调用者传递存储过程名和相应的存储过程参数，返回类型是 bool 类型。如果成功则执行该方法，并返回 true，否则返回 false。

6.4.2　任务 2：实现商品购买功能

1. 任务目标

（1）能掌握如何使用 URL 传值技术传递数据技术。

（2）能掌握 Request 对象 QueryString 属性。

（3）能掌握如何动态创建 DataTable 技术。

（4）能熟练运用 GridView 实现特殊效果。

2. 任务内容

（1）“购买”按钮用于实现导航和传值。

（2）“购物车”页面中获取“购买”按钮传递过来的数据。

（3）浏览器客户端实现加减操作。

（4）实现“更新数据”功能，更新购物车的数据。

（5）实现“继续购物”和“现在就去付款”导航功能。

3. 任务实施步骤

该任务首先是要实现商品显示页面中的“购买”功能。单击“购买”按钮(见图 6-10)跳转到“购物车”页面,并显示购物车信息(见图 6-11)。单击－按钮执行商品数量减 1 操作;单击＋按钮执行商品数量加 1 操作。单击“删除”链接执行删除购物记录功能。单击“更新数据”按钮,将购物车信息更新到 Session 中。单击“继续购物”按钮,跳转到商品显示页面。单击“现在就去付款”按钮,跳转到“订单生成”页面。



图 6-10　商品列表中的购买界面



图 6-11　“购物车”界面

(1) 双击商品列表显示文件 WareGrid,单击 GridView 控件,单击智能提示,单击“编辑模板”命令,打开“编辑模板”窗口。单击“购买”按钮的智能提示,单击“编辑 DataBinds”命令。单击 CommandArgument 属性,在代码表达式中输入 Eval("Ware_ID"),将“购买”按钮的 CommandArgument 属性动态绑定到数据字段 Ware_ID 中。

(2) 新建 ShoppingCart2. aspx 内容页(来自母版页 Layout. master)。拖动 GridView 控件到页面,更改其 ID 为 gvCart。

（3）双击“购买”按钮，进入“购买”按钮单击事件的相应代码，输入下列代码。

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    ImageButton btn= (ImageButton)sender;
    String Ware_ID= btn.CommandArgument;
    GridViewRow row= (GridViewRow)btn.NamingContainer;
    Label lblNum= (Label)row.Cells[1].FindControl("Label1");
    String addr= "~ /Shop/ShoppingCart2.aspx?wid= {0}&wnum= {1}";
    addr= String.Format(addr, Ware_ID, lblNum.Text);
    Response.Redirect(addr);
}
```

这段代码表示从“购买”按钮的单击事件传递的参数 sender 中获得触发事件的图像按钮 btn，然后从 btn 这个对象中获得步骤 1 中绑定的 CommandArgument 属性，也就是动态绑定的数据字段 Ware_ID 的值。接着通过 btn 的属性 NamingContainer 获得该按钮所在的行，从而根据 FindControl 方法获得“商品编号”标签的值。通过 Response 对象的 Redirect 方法跳转到 ShoppingCart2. aspx 页面，在跳转过程中携带了 wid（商品 ID）和 wnum（商品编码）值。

（4）双击 ShoppingCart2. aspx 页面，输入下列代码。

```
int ID= - 1;
String ProductNO= "";
SqlHelper sh= new SqlHelper();
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {
        try
        {
            ID= Convert.ToInt32(Request.QueryString["wid"].ToString());
            ProductNO= Request.QueryString["wnum"].ToString();
            BindList();
        }
        catch(Exception)
        {

            throw;
        }
    }
}
public void BindList()
{
    DataTable dt= new DataTable();
    if(Session["cart"] != null)
    {
        dt= (DataTable)Session["cart"];
    }
```

```
else
{
    dt.Columns.Add(new DataColumn("ID", typeof(Int32)));
    dt.Columns.Add(new DataColumn("ProductNo", typeof(String)));
    dt.Columns.Add(new DataColumn("ProductName", typeof(String)));
    dt.Columns.Add(new DataColumn("BuyPrice", typeof(Int32)));
    dt.Columns.Add(new DataColumn("Amount", typeof(Int32)));
}
//查询商品信息
if(ID != -1)
{
    bool isExisted= false;
    //判断是否存在
    foreach(DataRow dr in dt.Rows)
    {
        if(dr["ProductNo"].ToString().Trim()==ProductNO)
        {
            isExisted= true;
            break;
        }
    }

    if(!isExisted)
    {
        SqlParameter[] p= new SqlParameter[1];
        p[0]= new SqlParameter();
        p[0].ParameterName= "@Ware_ID";
        p[0].Value= ID;
        p[0].SqlDbType= SqlDbType.Int;
        DataTable dtproduct= sh.QueryOperationProc("sp_select2_id", p);

        dt.Rows.Add(new object[]{
            ID,
            dtproduct.Rows[0]["Ware_Number"].ToString(),
            dtproduct.Rows[0]["Ware_Name"].ToString(),
            Convert.ToInt32(dtproduct.Rows[0]["Ware_Price"]),
            1});
    }
    else
    {
        ScriptManager.RegisterStartupScript(this,typeof(Page),
        "alertExist","alert('您选择的商品(编号:"+ ProductNO+ ")
        已在购物车!')",true);
    }

}
gvCart.DataSource= dt;
gvCart.DataBind();
```

```
    Session.Add("cart", dt);
}
```

在 Page_Load 事件中的第一次加载过程中，通过 Request 对象的 QueryString 属性获得从“购买”按钮传递过来的 wid 和 wnum 值，然后调用 BindList 方法显示购物车信息。BindList 方法首先通过 if (Session["cart"] != null)语句来判断该用户对应的购物车是否存在(因为购物车信息是暂时存放在用户的 Session 对象中)，如果该用户购物车不存在，则通过 Dt. columns. add 方法创建购物车表，用于存放购买的商品信息，否则从 Session 对象中直接获取购物车表。接着使用 foreach (DataRow dr in dt. Rows)循环判断传递过来的 wnum 值是否已经在购物车表中，如果存在，则弹出提示信息(见图 6-12)，否则使用 SqlHelper 类的 QueryOperationProc 方法来调用 sp_select2_id 存储过程以获得给定的商品 ID 的商品具体信息，并通过 dt. Rows. Add 方法将查询到的商品具体信息添加到购物车表中。最后将购物车表的信息绑定到 GridView 控件中。



图 6-12　弹出提示信息

(5) 登录 SQL Server 2008 服务器，单击“新建查询”按钮，输入下列 SQL 代码，按 F5 功能键运行代码。新手容易出错的是选择了数据库 Smart。默认数据库是 master。

```
create proc sp_select2_id
@Ware_ID int
as
select Ware_Number, Ware_Name,Ware_Price from T_Ware where Ware_ID=@Ware_ID
```

(6) 单击 GridView 控件智能提示，再单击“自动套用格式”菜单项，打开“自动套用格式”对话框，单击“彩色型”项，再单击“确定”按钮。

(7) 依次单击 GridView 控件智能提示，单击“编辑列”菜单项，再单击 BoundField 项，单击“添加”按钮。单击 DataField 属性，输入 ProductNo；单击 HeaderText 属性，输入“商品

编号”。单击 BoundField 选项,单击“添加”按钮,单击 DataField 属性,输入 ProductName;单击 HeaderText 属性,输入“商品名”。单击 BoundField 选项,单击“添加”按钮,单击 DataField 属性,输入 BuyPrice。单击 DataFormatString,输入{0:C},确保 BuyPrice 列数据以货币符号格式显示。单击 HeaderText 属性,输入“单价”。单击 TemplateField 选项,单击“添加”按钮,单击 HeaderText 属性,输入“数量”。单击 CommandField→“删除”命令;再单击“添加”按钮;最后单击“确定”按钮。

(8) 单击 GridView 控件智能提示,单击“编辑模板”菜单项,打开“编辑模板”界面。拖动 HTML 控件箱中的 Image 控件到“编辑模板”界面中,单击 ID 属性,输入 imgReduce。单击 Src 属性,单击右边的“选择”按钮,设置图像文件为 WebIcon 中的 bg_close. gif。拖动 HTML 控件箱中的 Image 控件到“编辑模板”界面中。单击 ID 属性,输入 imgPlus。单击 Src 属性,单击右边的“选择”按钮,设置图像文件为 WebIcon 中的 bg_open. gif。拖动 TextBox 控件到 imgReduce 右边,单击 ID 属性,输入 txtAmount,单击智能提示,再单击“编辑 DataBinds”菜单项。单击 Text 属性,在代码表达式中输入 Eval("Amount"),将 txtAmount 文本框的 Text 属性绑定为数据字段 Amount。

(9) 按 F5 功能键运行程序。

6.4.3 任务 3:在客户端实现商品数量增减的功能

1. 任务目标

能使用 JavaScript 脚本编写客户端程序。

2. 任务内容

使用 JavaScript 脚本实现商品数量增减的功能。

3. 任务实施步骤

该任务首先需要在页面中使用 JavaScript 添加“加”和“减”函数,然后在 GridView 的 RowDataBound 事件中给 imgReduce 图像和 imgPlus 图像分别添加客户端代码。

(1) 按 Shift+F7 组合键切换到页面。单击“源”按钮,切换到“源”视图,在<asp:content>后输入 Reduce 和 Plus 函数。

```
<script type="text/javascript">
    function Reduce(obj) {
        if(obj.value>1) {
            obj.value=obj.value - 1;
        }
    }
    function Plus(obj) {
        obj.value=parseInt(obj.value)+1;
    }
</script>
```

(2) 单击“设计”按钮;再单击 GridView 控件;接着单击“事件”按钮,双击 RowDataBound 事件,进入代码编辑状态,输入下列代码:

```
protected void gvCart_RowDataBound(object sender, GridViewRowEventArgs e)
{
```

```
        if(e.Row.RowType==DataControlRowType.DataRow)
        {
            e.Row.Attributes.Add("onmouseover",
              "b=this.style.backgroundColor;this.style.backgroundColor='#E1ECEE'");
            e.Row.Attributes.Add("onmouseout",
              "this.style.backgroundColor=b");
            TextBox tb=(TextBox)e.Row.FindControl("TxtAmount");
            ((HtmlImage)e.Row.FindControl("imgReduce")).Attributes.Add("onclick",
              "Reduce("+tb.ClientID+")");
            ((HtmlImage)e.Row.FindControl("imgPlus")).Attributes.Add("onclick",
              "Plus("+tb.ClientID+")");
        }
    }
```

通过 if (e. Row. RowType==DataControlRowType. DataRow)语句判断当前行是否是数据行。在数据行内通过 FindControl 查找 txtAmount 文本框;再查找 imgReduce 图像并转换为 HtmlImage 类型;接着给该对象的 Attributes 属性添加 onClick 事件名和对应的 Reduce 函数。以同样方式给 imgPlus 的 Attributes 属性添加 onClick 事件名和对应的 Plus 函数。

(3) 按 F5 功能键执行程序,效果如图 6-13 所示。



图 6-13　任务 3 效果图

6.4.4　任务 4:删除购物车中的记录

1. 任务目标

能使用 DataTable 类删除数据。

2. 任务内容

删除购物车中的记录。

3. 任务实施步骤

单击 GridView 控件,再单击"事件"按钮,然后双击 RowDeleting 事件,进入代码页面,输入下列代码:

```
protected void gvCart_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    DataTable dt= Session["cart"] as DataTable;
    dt.Rows.RemoveAt(e.RowIndex);
    dt.AcceptChanges();
    Session["cart"]= dt;
    gvCart.DataSource= dt;
    gvCart.DataBind();
}
```

以上代码片段表示从 Session 对象中获取 cart 属性值并转换成 DataTable 类型，接着将“删除”按钮触发的行记录删除，最后重新将 gvCart 重新绑定数据。原先购物记录有 2 条，单击“删除”按钮后只剩 1 条记录，效果如图 6-14 所示。



图 6-14　删除功能

6.4.5　任务 5：订单的生成

1. 任务目标

(1) 能使用 JavaScript 脚本实现特殊效果。

(2) 能使用事务技术保持数据库操作的原子性。

2. 任务内容

(1) 使用 JavaScript 显示已经隐藏的 DIV。

(2) 使用 ADO. NET 级事务执行插入订单的操作。

3. 任务实施步骤

该任务的重点是使用 ADO. NET 级事务执行插入订单操作。基本操作流程是：从购物车的“现在就去付款”按钮跳转到订单生成页面(见图 6-15，其前提是已经登录过，否则会跳转到登录页面并要求用户进行登录)。如果地址栏中没有所需的地址，则单击地址栏最右边的展开按钮，显示“新地址”一栏，在新地址文本框中输入地址，单击“＋”按钮，添加新地址到 T_CustomerAddress 地址中。单击“去结算”按钮，生成订单并跳转到商品列表页面。

(1) 从 Layout. master 生成内容页 Order. aspx，并按照图 6-16 所示设计界面。读者可以使用 Table 控件或者 DIV 控件进行布局。页面的 HTML 代码如下所示。



图 6-15　订单生成页面

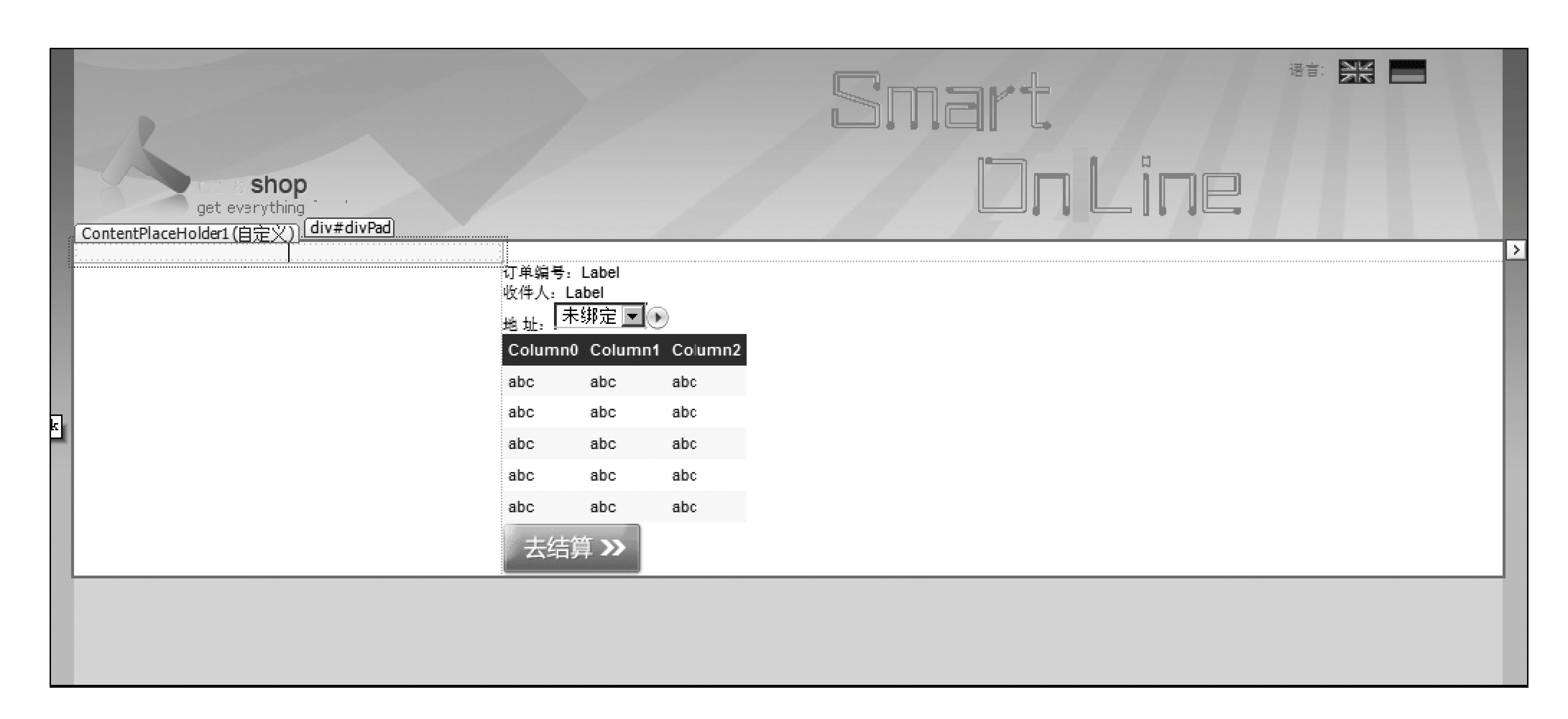


图 6-16　任务 5 界面布局

```
<%@ Page Title="" Language="C#"  MasterPageFile="~/Shop/Layout.master"
    AutoEventWireup="true" CodeFile="Order.aspx.cs" Inherits="Shop_Order" %>
    <asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
    Runat="Server">
 <link href="mystyle.css" rel="stylesheet" type="text/css" />
  <div id="divPad">
  </div>
    <div id="divOrderMain">
      <div>订单编号：<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </div>
      <div id="divuserinfo">
            收件人：<asp:Label ID="lblUser" runat="server" Text="Label"></asp:Label>
      </div>
    <div id="divaddress">
           地　址：<asp:DropDownList ID="ddlAddress" runat="server"
           DataTextField="Customer_Address" DataValueField="CustomerAddress_ID">
```

```
                <asp:DropDownList>
                <img id="imgExtend" alt="" src="../WebIcon/16/play.png"
                    style="vertical-align:middle" runat="server"/></div>
                <div id="divadd" runat="server" style="display:none">新地址：
                        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:ImageButton ID="ImageButton2" runat="server"
                        ImageUrl="~ /WebIcon/16/add.png"  style="vertical-align:middle"
                        OnClick="ImageButton2_Click"/>
                </div>
            <div id="divproinfo">
                <asp:GridView ID="gvInfo" runat="server" CellPadding="4"
                ForeColor="#333333" GridLines="None">
                    <AlternatingRowStyle BackColor="White" />
                    <FooterStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
                    <HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
                    <PagerStyle BackColor="#FFCC66" ForeColor="#333333"
                            HorizontalAlign="Center" />
                    <RowStyle BackColor="#FFFBD6" ForeColor="#333333" />
                    <SelectedRowStyle BackColor="#FFCC66" Font-Bold="True"
                        ForeColor="Navy" />
                    <SortedAscendingCellStyle BackColor="#FDF5AC" />
                    <SortedAscendingHeaderStyle BackColor="#4D0000" />
                    <SortedDescendingCellStyle BackColor="#FCF6C0" />
                    <SortedDescendingHeaderStyle BackColor="#820000" />
                </asp:GridView>
            </div>
            <div id="divsubmit">
                <asp:ImageButton ID="ImageButton1" runat="server"
                    ImageUrl="~/WebIcon/pay.gif" onClick="ImageButton1_Click" />
            </div>
            </div>
        </asp:Content>
```

（2）页面加载是自动绑定订单号、用户账号和地址、购物车信息。按 F7 功能键切换到代码页面，在 Page_Load 事件中输入下列代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    if(Session["useraccount"]==null)
    {
        Response.Redirect("~/Shop/Login.aspx");
    }
    else
    {
        lblUser.Text=Session["useraccount"].ToString();
        if(!Page.IsPostBack)
        {
            ((HtmlImage)Master.FindControl("ContentPlaceHolder1").
                FindControl("imgExtend")).Attributes.Add("onclick","DisplayAddr()");
```

```
                if(Session["cart"] != null)
                {
                    DataTable dt= Session["cart"] as DataTable;
                    gvInfo.DataSource= dt;
                    gvInfo.DataBind();
                }
                Label1.Text= Convert.ToString(DateTime.UtcNow.Millisecond);
                String sql= String.Format("select customer_id from T_Customer where
                    Customer_Account= '{0}'", lblUser.Text);
                SqlDataReader sdr= sh.QueryOperation(sql);
                int customerid= - 1;
                if(sdr.HasRows)
                {
                    sdr.Read();
                    customerid= sdr.GetInt32(0);
                }
                String sql3= String.Format("select * from T_CustomerAddress
                    where customer_id= {0}", customerid);
                SqlDataReader sdr2= sh.QueryOperation(sql3);
                ddlAddress.DataSource= sdr2;
                ddlAddress.DataBind();
            }
        }
    }
```

该代码片段首先通过 Session["useraccount"] == null 判断用户是否登录过，若没有登录过，则跳转到登录页面；否则执行下面的逻辑：先通过((HtmlImage) Master. FindControl("ContentPlaceHolder1")。

FindControl("imgExtend"))方法查找页面中的“展开”图像框的 imgExtend 控件，然后通过 Attributes 属性的 add 方法给其添加 onClick 事件和逻辑代码 DisplayAddr()。这里需要注意的是，如果套用了母版页，则必须先使用 Master. FindControl("ContentPlaceHolder1")查找母版页中的 ContentPlaceHolder 控件，然后再采用 FindControl 查找其他控件；再将 Session 中的购物车信息绑定到 gvInfo 控件；然后调用 SqlHelper 类的 QueryOperation 方法查询会员账号对应的会员 ID；最后根据会员 ID 号查询该会员所对应的地址并绑定到 ddlAddress 下拉列表框。

(3) 按 Shift+F7 组合键切换到页面，单击“源”按钮切换到 HTML 视图，在<asp:content>后面输入 DisplayAddr()函数。

```
<script type= "text/javascript">
  unction DisplayAddr() {
    document.getElementById("<%=divadd.ClientID%>").style.display= "block";
  }
</script>
```

若要显示某个 HTML 控件，需将 HTML 控件的 Style 的 Display 属性设置为 block；若要隐藏 HTML 控件，则设置其为 none。

（4）编写添加地址的功能。双击“+”按钮，输入下列代码。

```
protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
{
try
    {
        String sql=String.Format("select customer_id from T_Customer
            where Customer_Account='{0}'",lblUser.Text);
        SqlDataReader sdr  =sh.QueryOperation(sql);
        int customerid=-1;
        if(sdr.HasRows)
        {
            sdr.Read();
            customerid=sdr.GetInt32(0);
        }
        String sql2=String.Format("insert into T_CustomerAddress values({0},
              '{1}')",
              customerid, TextBox1.Text);
        sh.ExeNonQuery(sql2);
        String sql3=String.Format("select * from T_CustomerAddress where customer_id
              ={0}", customerid);
        SqlDataReader sdr2=sh.QueryOperation(sql3);
        ddlAddress.DataSource=sdr2;
        ddlAddress.DataBind();
    }
    catch(Exception)
    {

        throw;
    }
}
```

（5）双击“去结算”按钮，切换到代码视图，在单击事件代码段中输入如下代码。

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    SqlConnection con=sh.getConn();
    con.Open();
    SqlTransaction trans=con.BeginTransaction();
    try
    {
        String sql=String.Format("select customer_id from T_Customer
            where Customer_Account='{0}'", lblUser.Text);
        SqlCommand cmd=new SqlCommand(sql, con);
        cmd.Transaction=trans;
        SqlDataReader sdr=cmd.ExecuteReader();
        int customerid=-1;
        if(sdr.HasRows)
        {
            sdr.Read();
```

```
                customerid= sdr.GetInt32(0);
            }
            sdr.Close();
            int CustomerAddressID= -1;
            CustomerAddressID= Convert.ToInt32(ddlAddress.SelectedValue.ToString());
            String sql2= String.Format("insert into T_Order
                values('{0}',{1},{2},'{3}',{4},{5},'{6}');
                select cast(scope_identity() as int)",
                Label1.Text, customerid, CustomerAddressID, "货到付款",
                    0, 0, DateTime.Now);
            cmd.CommandText= sql2;
            int OrderID= (int)cmd.ExecuteScalar();
            for(int i= 0; i < gvInfo.Rows.Count; i++)
            {
                int WareID= Convert.ToInt32(gvInfo.Rows[i].Cells[0].Text);
                float qty= Convert.ToInt32(gvInfo.Rows[i].Cells[4].Text);
                String sql3= String.Format("insert into T_ShoppingCart
                    values({0},{1},{2},{3})",
                    customerid,WareID,qty,OrderID);
                cmd.CommandText= sql3;
                cmd.ExecuteNonQuery();
            }
            trans.Commit();
        }
        catch(Exception ex)
        {
            trans.Rollback();
        }
        con.Close();
        Session["cart"]= null;
        Response.Redirect("~ /Shop/WareGrid2.aspx");
    }
```

生成订单的功能采用 ADO. NET 级事务技术以确保 T_Order 和 T_ShoppingCart 表数据操作同步，要么全都插入，要么全都不插入。这里要特别注意的是，当向 T_Order 表中插入一条订单记录后，需要记录这条新产生的订单记录的 ID，那怎么来获取刚插入的订单 ID 呢? 细心的读者可能发现了下面粗体字的内容。

```
String sql2= String.Format("insert into T_Order values('{0}',{1},{2},'{3}',{4},{5},'{6}');
select cast(scope_identity() as int)"
```

scope_identity ()用于传回目前工作阶段任何表中所产生的最后一个序列值。这里将 insert 语句和 scope_identity 连用，可以返回刚插入的 T_Order 表的序列值。这个序列值是插入 T_ShoppingCart 表中的记录时所需要的。

6.5 总 结 归 纳

子项目 6 主要是完成了购物功能，使用了事务、DataTable 类、存储过程和 GridView 高

级技巧等技能。事务实现方式通常分为 ADO.NET 级别事务、ASP.NET 页面级事务和 System.Transactions 操作事务。对于通过 ADO.NET 访问数据库方面的基本知识，由于篇幅关系，本项目中不再细述，若有疑问，可以参阅项目 3 中的 ADO.NET 技术的介绍。

6.6 课后习题

选择题

1. 一个 DataSet 可以包含(　　)数据表(DataTable)。

A. 1 个　　B. 2 个　　C. 3 个　　D. 多个

2. 一个 DataTable 可以动态生成(　　)DataView。

A. 1 个　　B. 2 个　　C. 3 个　　D. 多个

3. 下列属于强类型的是(　　)。

A. ArrayList 对象　　B. DataTable 对象　　C. DataView 对象　　D. 实体对象

4. 在 ADO.NET 中，为访问 DataTable 对象从数据源提取的数据行，可使用 DataTable 对象的(　　)属性。

A. Rows　　B. Columns　　C. Constraints　　D. DataSet

5. 已知变量 ds 引用某个 DataSet 对象，该 DataSet 对象中已包含一个表名为 table1 的数据表。在 Windows 窗体 Form1 中，为了将变量名为 dgvData 的 DataGridView 控件绑定到数据表 table1 中，可以使用代码( )。(选两项)

A. dgvData.DataSource=ds;
　dgvData.DataMember=ds.Tables["table1"];

B. dgvData.DataMember=ds;

C. dgvData.DataSource=new DataView(ds.Tables["table1"]);

D. dgvData.DataSource=ds.Tables["table1"];
　dgvData.DataMember=ds;

6. 对事务描述错误的是(　　)。(选两项)

A. 一个事务中的所有命令会作为一个整体提交或回滚

B. 如果两个并发事务要同时修改同一个表，有可能产生死锁

C. SQL Server 默认将每条单独的 T-SQL 语句视为一个事务

D. 必须使用 begin transaction 来明确指定事务的开始

7. 在 SQL Server 2008 中，创建如下存储过程。要在 Students 表中查找 Age(年龄)是 18 岁的学生，(　　)语句可以正确地调用这个存储过程。(选两项)

```
CREATE PROCEDURE MyP1 @p Int AS
SELECT Studentname,Age FROM Student WHERE Age=@p
```

A. EXEC Myp1 18　　B. EXEC Myp1 @p=18

C. EXEC Myp1 p='18'　　D. EXEC Myp1 p=18

8. 分析下面的存储过程。

```
CREATE PROCEDURE MyP1
(
  @a varchar(32)
)
AS
BEGIN tran
DECLARE @b int
DELETE FROM a1 WHERE au_lname LIKE @a
SELECT @b=@@rowcount
if(@@error!=0)
  BEGIN
    ROLLBACK tran
    RETURN 200
  END
DELETE FROM a2 WHERE au_lname LIKE @a
SELECT @b=@b+@@rowcount
IF(@@error!=0)
  BEGIN
    ROLLBACK tran
    RETURN  200
  END
COMMIT tran
RETURN @b
```

下面选项正确的有(　　)。(选两项)

A. 该存储过程是无效的,也不会被创建

B. 如果在 a1 表的删除操作中发生错误,那么它在 a2 表中就不会执行删除操作

C. 如果在 a2 表中执行删除操作时发生错误,那么 a1 表中删除的行就会被回滚回去

D. 存储过程会成功执行,并返回 200

6.7 同步操练

格林酒店管理系统要求支持前台客房预订。当前台接到预订电话后,需要在系统中添加客房预定信息(图 6-17)。当单击"日期"文本框时,会自动弹出日期来让用户进行选择(图 6-18),当选择好客房类型后,单击"搜索"按钮会显示符合要求的房间号(图 6-19)。当然也需要提供预订编辑功能(图 6-20)。



图 6-17　添加预订功能



图 6-18　日期的选择



图 6-19　搜索符合条件的房间

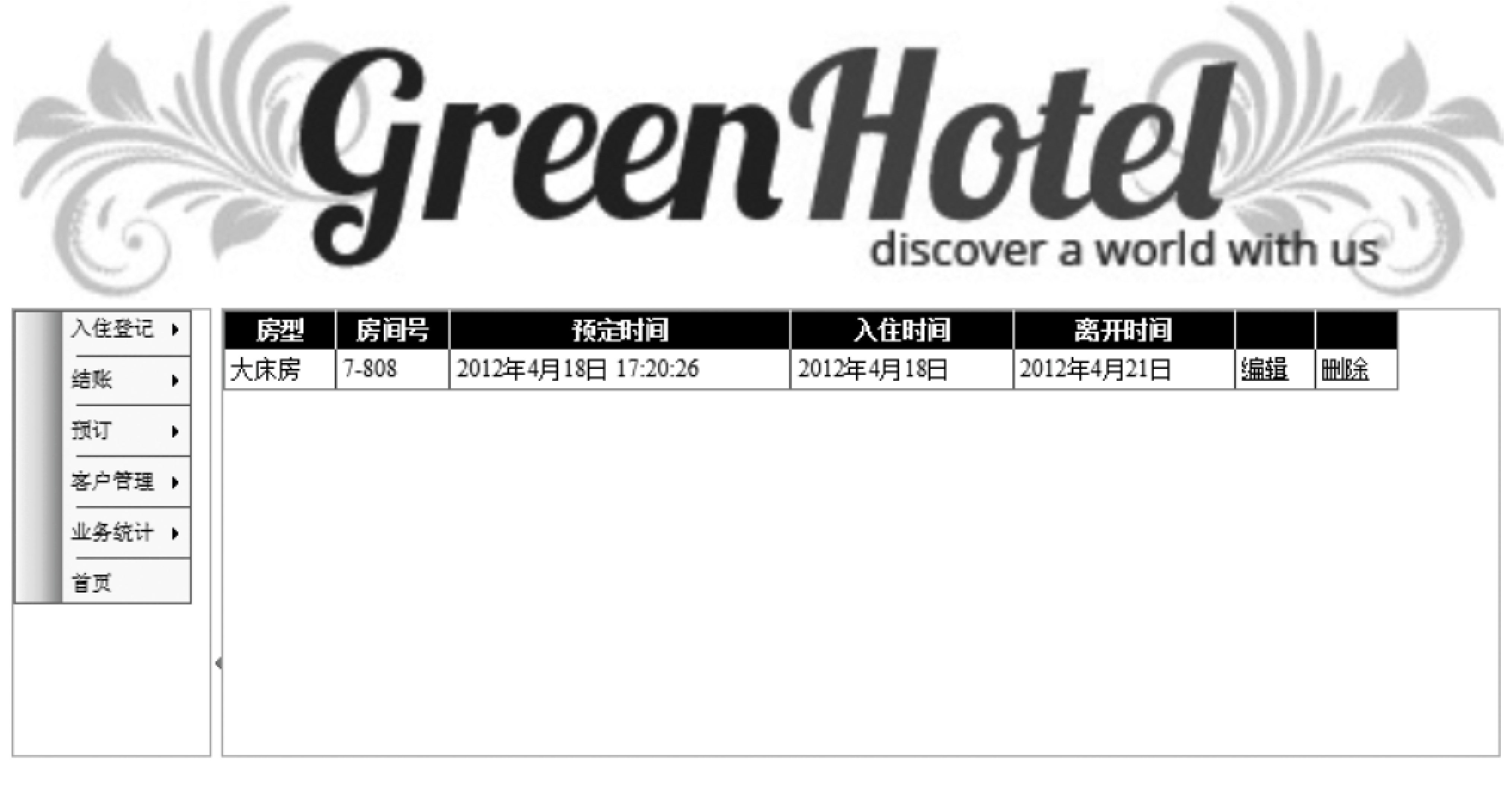


图 6-20　“预订”信息的编辑

项目 7　商品信息管理

7.1　项目引入

前面项目中经常使用商品信息等数据,那这些数据是怎么来的呢?其实是系统管理人员通过后台中的商品信息管理模块添加和编辑所得到的。李明是 Smart On Line 电子商城的一位开发人员,接到的任务是协助创建商品信息管理模块,具体包括商品的一级目录、二级目录、三级目录和商品信息的添加和编辑等功能。

7.2　项目分析

经过小组讨论和仔细分析,采用 ADO. NET 方式来完成一级目录、二级目录、三级目录和商品信息的添加与编辑。项目实现的难点在于商品信息的管理,不同的商品除了具有共同的一些信息外,还具有其特有的信息。例如手机具有的特有信息和衣服类具有的特有信息就不一样。所以小组在开发商品信息添加功能的时候,需要动态创建子类型表来保存这些特有信息。商品信息里必须具备相应的商品图像,该商品图像需要通过 FileUpLoad 控件上传图像到服务器上。

7.3　知识准备

7.3.1　使用 FileUpLoad 上传文件

应用程序经常需要用户把文件上传到 Web 服务器。Visual Studio 2012 提供了 FileUpLoad 控件来帮助开发人员实现该功能。该控件让用户更容易地浏览和选择用于上传的文件,它包含一个浏览按钮和用于输入文件名的文本框。只要用户在文本框中输入了完全限定的文件名,无论是直接输入或通过“浏览”按钮选择,都可以调用 FileUpload 的 SaveAs 方法将该文件保存到磁盘上。除了从 WebControl 类继承的标准成员,FileUpload 控件还公开了几个只读的属性,如表 7-1 和表 7-2 中所示。

表 7-1　FileUpload 控件的属性

名　称	类　型	读	写	说　明
FileContent	Stream	×		返回一个指向上传文件的流对象
FileName	string	×		返回要上传文件的名称，不包含路径信息
HasFile	Boolean	×		如果是 true，则表示该控件有文件要上传
PostedFile	HttpPostedFile	×		返回已经上传文件的引用。表 5-9 列出了它所公开的只读属性

表 7-2　HttpPostedFile 属性

名　称	类　型	读	写	说　明
ContentLength	integer	×		返回上传文件的按字节表示的文件大小
ContentType	string	×		返回上传文件的 MIME 内容类型
FileName	string	×		返回文件在客户端的完全限定名
InputStream	Stream	×		返回一个指向上传文件的流对象

使用 FileUpload 的步骤一般如下。

① 选择要上载的文件。

② 显式提供一个控件或机制，使用户能提交指定的文件。例如，单击按钮来上载文件。

③ 使用 HasFile 属性来验证 FileUpload 控件确实包含了文件。

④ 调用 SaveAs 方法将文件内容保存到服务器上的指定路径中。

调用 SaveAs 方法时，必须指定用来保存上载文件的目录的完整路径。如果没有在应用程序代码中显式指定路径，则当用户试图上载文件时将引发异常。使用 FileName 属性可以获取客户端上使用 FileUpload 控件上载的文件的名称。此属性返回的文件名不包含此文件在客户端上的路径。常使用 Server. MapPath 方法获得虚拟目录下的完整路径。例如：

```
string path= Server.MapPath("~/Temp/")
```

ASP. NET 默认会对上传文件的大小进行限制，这点可以在配置文件(web. config)中配置。

```
<system.web>
    <httpRuntime executionTimeout= "300"
        maxRequestLength= "40960" useFullyQualifiedRedirectUrl= "false"/>
</system.web>
```

maxRequestLength 表示可上传文件的最大值，一旦超过这个设置值，就不能通过 FileUpLoad 将文件上传到服务器。

ContentType 获取上传文件的 MIME 内容类型。PostedFile. ContentType 这个属性用来判断上传文件的 MIME 类型，通常以此来拒绝某些类型的上传，比如仅允许上传图像文件，或者指定固定图片的格式，常见的图片 MIME 类型为 image/jpeg、image/png 等。

示例 1：单击“上传”按钮，将选择的图片上传到站点根目录中的 Photo 文件夹，并在 Image 控件中显示该图片，如图 7-1 所示。

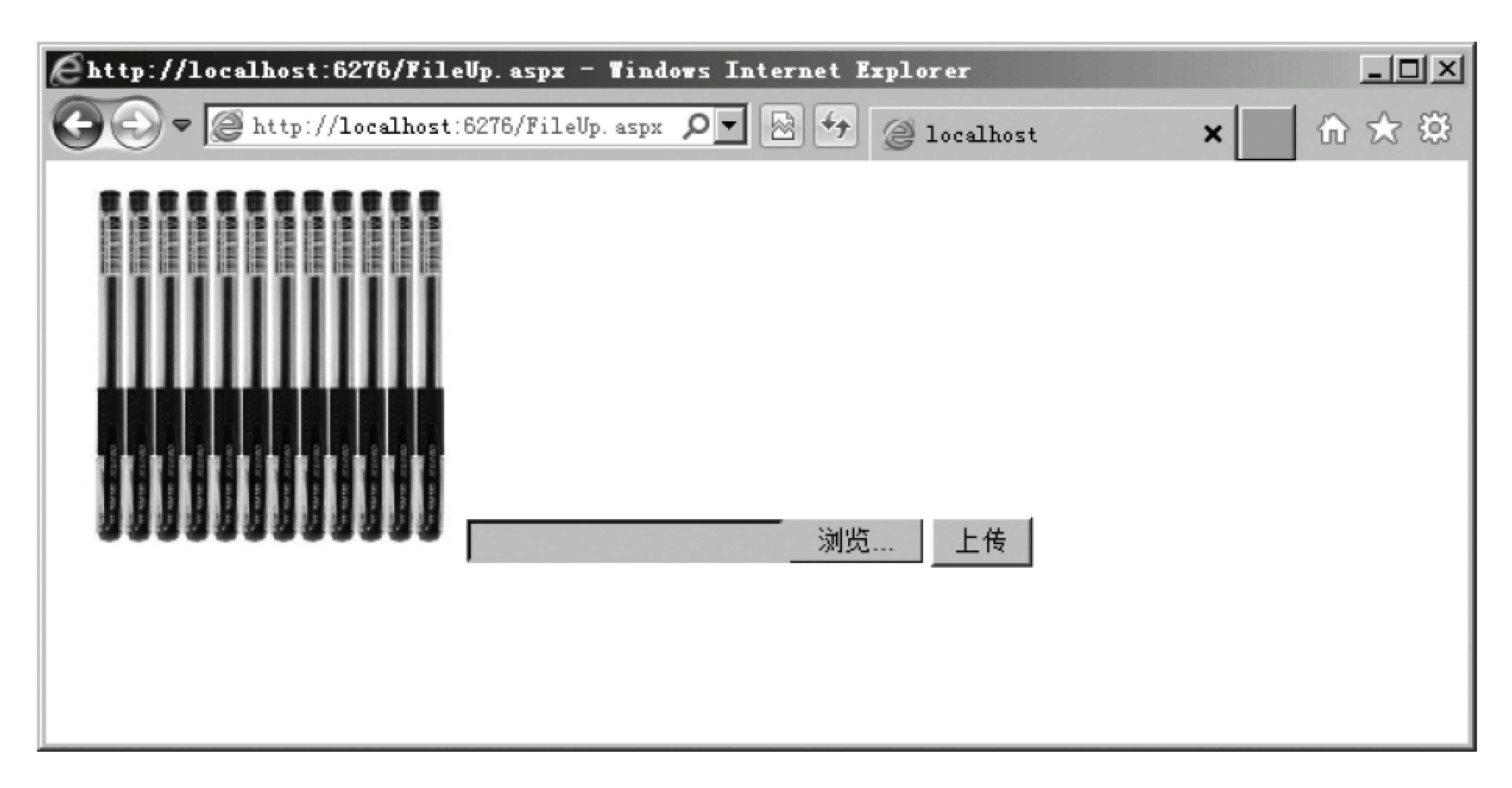


图 7-1　示例 1 效果

① 新建网站。具体操作这里不再重述,假设网站名为 Ex7-1。

② 右击 Ex7-1 文件夹,单击"添加"→"添加新项"命令,打开"添加新项"对话框。单击"Web 窗体"列表项,在"名称"文本框中输入 FileUp.aspx。

③ 拖动 Image 控件到页面上,单击 Height 属性,输入 190px;单击 Width 属性,输入 180px。

④ 拖动 FileUpLoad 控件到 Image 控件的右边,拖动 Button 控件到 FileUpLoad 控件的右边。

⑤ 右击 Ex7-1 文件夹,单击"添加"→"新建文件夹"命令,输入 Photo。双击"上传"按钮,进入事件代码编辑状态,输入下列代码。

```
protected void btnupload_Click(object sender, EventArgs e)
{

    int length=this.FileUpload1.PostedFile.ContentLength;
    if(length>102400)
    {
        Response.Write("<script language='javascript'>alert('您选择的图片过大!
        ');</script>");
    }
    else{
    String type=this.FileUpload1.PostedFile.ContentType;
    String fullfilename=this.FileUpload1.PostedFile.FileName;
    String filename=fullfilename.Substring(fullfilename.LastIndexOf("\\")+1);
    String extensions=filename.Substring(filename.LastIndexOf(".")+1);
    string name=DateTime.Now.ToString("yyyyMMddHHmmss");
    if(type=="image/jpeg" || type=="image/png")
    {
        this.FileUpload1.SaveAs(Server.MapPath("Photo")+"\\"+
            name+"."+extensions);
        this.Image1.ImageUrl="Photo/"+name+"."+extensions;
    }
    else
```

```
        {
            Response.Write("<script language='javascript'>alert('您选择的图片有误!');</script>");
        }
    }
```

“上传”按钮的单击事件的相应代码中，首先通过 this. FileUpload1. PostedFile . ContentLength 获取上传文件的长度，如果大于 102 400 字节，则通过 Response. Write 弹出提示信息。如果符合要求，再通过 String type＝this. FileUpload1. PostedFile. ContentType 语句来判断文件内容，如果是 jpg 或者 png 文件格式，则以当前时间戳为文件名进行命名并保存到 Photo 文件夹中。但是网站中提供的是虚拟路径，需要使用 Server. MapPath ("Photo") 语句将其映射到物理路径中。

7.3.2　保存图片到数据库中

通常用户上传的图片需要保存到数据库中。解决方法一般有两种：一种是将图片保存的路径存储到数据库中，这种方法向数据库中保存的不是图片本身，而是图片的地址，读取的也是图片的地址，再根据保存的地址定位到指定的图片。另一种是将图片以二进制数据流的形式直接写入数据库字段中。

示例 2： 继续完善示例 1 的任务，单击“上传”按钮，以二进制数据流的形式直接将信息写入数据库。

① 创建数据库和相应表。登录到 SQL Server 2008 服务器，在“对象资源管理器”中右击“数据库”对象，单击“新建数据库”菜单项，在“数据库”文本框中输入 Test。

② 单击“新建查询”按钮，打开“查询分析”窗口，输入下列 SQL 代码，再按 F5 功能键执行代码并创建 ImgTest 表。

```
Create Table ImgTest
(
    IID int identity(1,1) primary key,
    info varchar(50),
    img Image
)
```

IID 是主键，自动编号；info 用来存储上传图片的格式；img 用来存储图片的二进制格式数据。

③ 双击“上传”按钮，进入代码编辑页面。在上传过程中需要将数据流以二进制形式保存到数据库中，涉及 ADO. NET 和输入/输出流，所以需要使用 using 指令导入命名空间。

```
using System.IO;
using System.Data.SqlClient;
using System.Data;
```

接着在 btnupload_Click 文件中输入下面代码中的粗体部分。

```
protected void btnupload_Click(object sender, EventArgs e)
{

    int length=this.FileUpload1.PostedFile.ContentLength;
```

```
        if(length>102400)
        {
            Response.Write("<script language='javascript'>alert('您选择的图片过大！
            ');</script>");
        }
        else{
            String type=this.FileUpload1.PostedFile.ContentType;
            String fullfilename=this.FileUpload1.PostedFile.FileName;
            String filename=fullfilename.Substring(fullfilename.LastIndexOf("\\")+1);
            String extensions=filename.Substring(filename.LastIndexOf(".")+1);
            string name=DateTime.Now.ToString("yyyyMMddHHmmss");
            if(type=="image/jpeg" || type=="image/png")
            {
                this.FileUpload1.SaveAs(Server.MapPath("Photo")+"\\"+name+"."
                +extensions);
                this.Image1.ImageUrl="Photo/"+name+"."+extensions;
                Stream imgStream=FileUpload1.PostedFile.InputStream;
                int imgSize=FileUpload1.PostedFile.ContentLength;
                byte[] imgContent=new byte[imgSize];
                imgStream.Read(imgContent, 0, imgSize);
                imgStream.Close();
                string strConn="server=.;database=test;user id=sa;password=zzb";
                SqlConnection conn=new SqlConnection(strConn);
                SqlCommand comm=conn.CreateCommand();
                String imgType=FileUpload1.PostedFile.ContentType;
                string sql="insert into imgtest(info,img) values(@info,@img)";
                comm.CommandText=sql;
                comm.Parameters.Add("@info", SqlDbType.VarChar, 50);
                comm.Parameters.Add("@img", SqlDbType.Image);
                comm.Parameters["@info"].Value=imgType;
                comm.Parameters["@img"].Value=imgContent;
                try
                {
                    conn.Open();
                    comm.ExecuteNonQuery();
                    conn.Close();
                }
                catch(Exception ex)
                {
                    Response.Write("<script language='javascript'>alert('
                    数据插入错误！');</script>");
                }
            }
            else
            {
                Response.Write("<script language='javascript'>alert('您选择的图片有误！');</
                script>");
            }
        }
    }
```

以上代码段通过 Stream imgStream＝FileUpload1. PostedFile. InputStream 获得上传文件的输入流，通过 Stream 对象的 read 方法将数据读入 imgContent 数组中，然后通过调用 ADO. NET 技术插入数据库。可见将图片上传到数据库的关键是将输入量转换为字节类型的数组。如果转换成功，将会在数据库中成功插入一条记录(见图 7-2)。

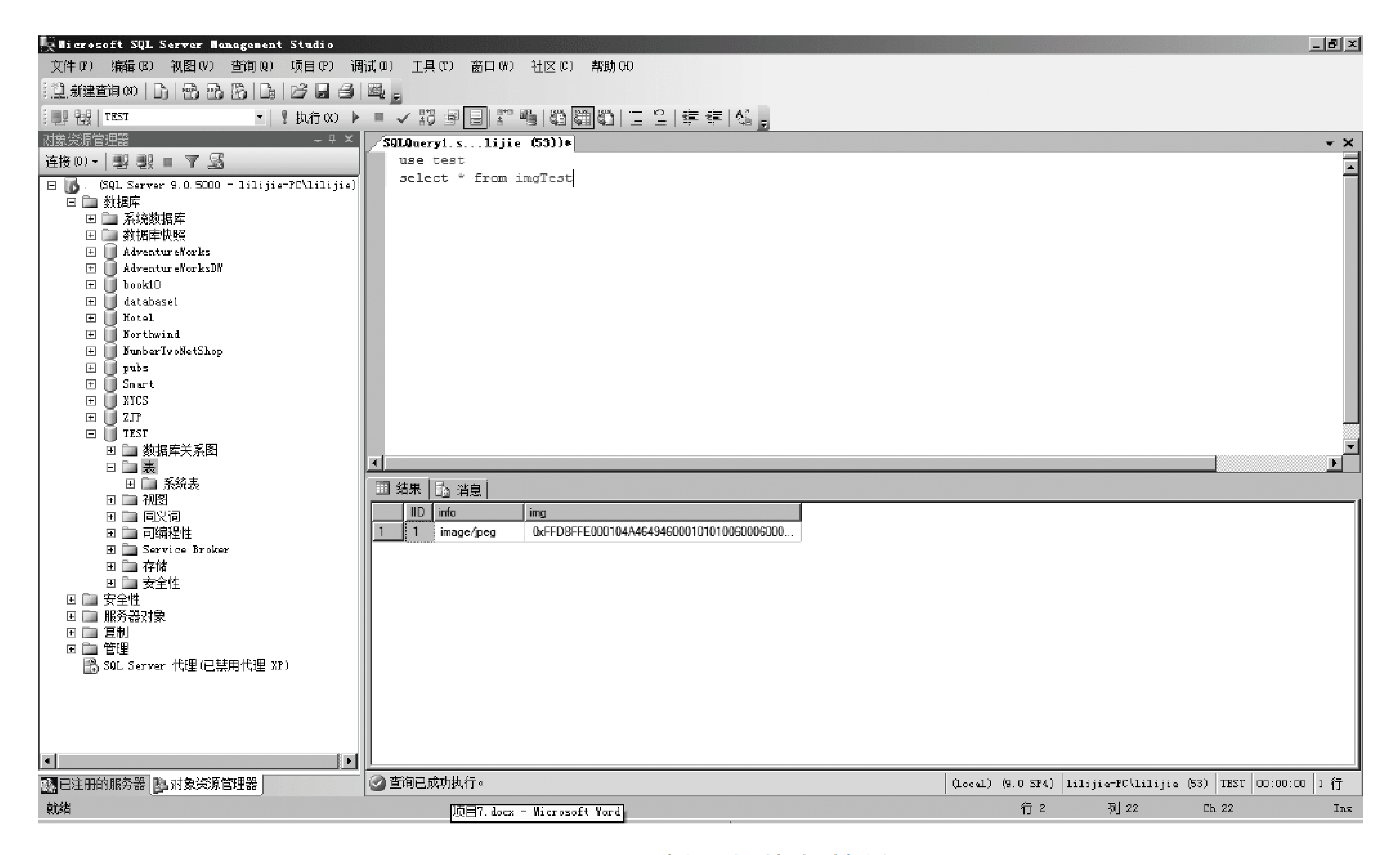


图 7-2　示例 2 的执行结果

7.3.3　GridView 模板列

关于模板列可以用四个字来描述——“无所不能”。很多棘手的问题，在用到模板列的时候会迎刃而解，因为模板列的定制功能很强大，允许开发人员建立不同状态下的模板，所以用代码控制起它来也就更加灵活自如了。创建模板列一般有两种方式，一是在“编辑列”对话框中向“选中列”添加 TemplateField 字段；二是在“编辑列”对话框中单击“选中列”中相应的列，再单击“确定”按钮上方的“将此字段转换为 TemplateField”文字，这样就把现有的列转换为模板列了。这种转换方式大家要小心使用，因为一旦转换为模板列，就没办法再转换回去了。特别要注意的是，当一个绑定列转换为模板列后，该列的数据显示时，会把数据源的数据绑定到模板列的 Label 控件中去。而不是像绑定列那样直接把数据绑定到单元格的 Text 属性上。所以取模板列的单元格中的值与取绑定列的单元格中的值是有区别的。取模板列单元格中的控件值先要采用 FindControl 方法查找对应的控件，然后才能访问对应控件的属性，如下面的代码所示(假如这里的控件是一个 Label 控件)。

```
((Label)(GridView.Rows[index].Cells[2].FindControl("ControlID"))).Text
```

而取绑定列单元格直接可以通过 e. Row. Cells[i]. Text 代码进行访问。

示例 3：如图 7-3 所示，在项目 5 的示例 8 的基础上添加 jobs 表的数据检查(当输入不符合要求的数据时出现 *)和更新功能(见图 7-4)，并完善删除功能。要求删除行记录前弹出提示信息(见图 7-5)。

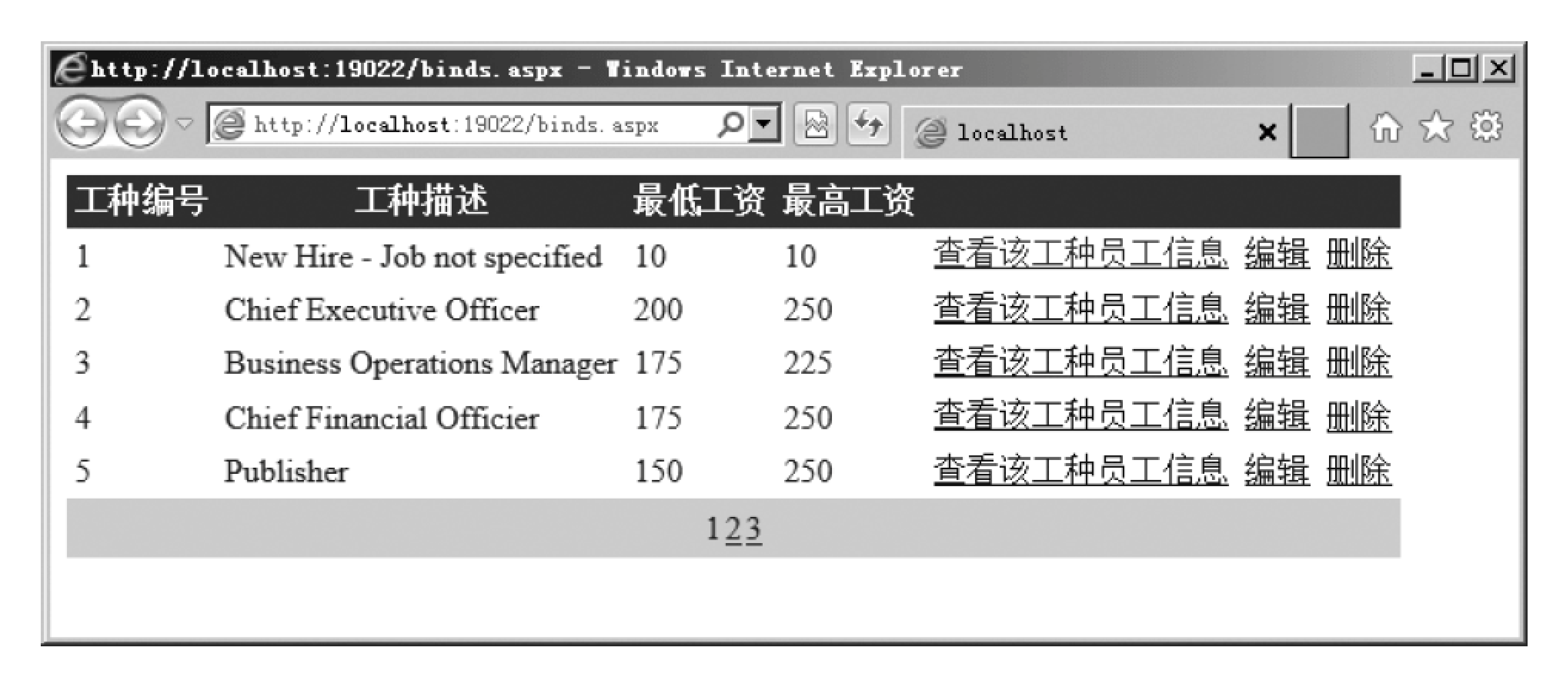


图 7-3　示例 3 界面

图 7-4　更新时做数据检查



图 7-5　删除操作时的提示信息

① 打开 Visual Stdio 2012 开发工具，单击"文件"→"打开"→"网站"菜单项，出现"打开网站"对话框，选择本书配套提供的 Ex5-1 网站，单击"打开"按钮。

② 在"解决方案资源管理器"窗口中双击 Binds.aspx 文件。根据图 7-4 所示，"工种编号"和"工种描述"列在编辑状态是不可编辑的，也就是说这两个列是只读的，需要设置这两列。单击 GridView 控件，再单击智能提示，然后单击"编辑列"菜单项，打开"编辑列"对话

框。单击“工种编号”列，将其 Readonly 属性设置为 true。同样，单击“工种描述”列，将其 Readonly 属性设置为 true。

③ 在“编辑列”对话框中单击“最低工资”列，单击右下方的“将此字段转换为 TemplateField”选项，将“最低工资”列转换为模板列。单击“最高工资”列，单击右下方的“将此字段转换为 TemplateField”选项，将“最高工资”列转换为模板列。单击“删除”命令列，单击右下方的“将此字段转换为 TemplateField”选项，将“删除”命令列转换为模板列。单击 CommandField→“编辑、更新、取消”命令，单击“添加”按钮。

④ 单击 GridView 右上角的“智能提示”→“编辑模板”菜单项，打开“编辑模板”窗口。单击“删除”按钮，单击“智能提示”→“编辑 DataBinds”菜单项，打开“编辑 DataBinds”窗口，再单击 CommandArgument 属性，在“绑定表达式”文本框中输入 Eval("job_id")，单击“确定”按钮。单击“删除”按钮，在“属性”窗口中单击 OnClientClick 属性，输入代码“return confirm('确认删除吗')?”，完成了删除操作的确认，“删除”按钮所对应的 RowDeleting 事件在项目 5 的示例 8 中已经实现，所以这个例子中不再重复演示。

⑤ 单击 GridView 右上角的“智能提示”→“编辑模板”菜单项，打开“编辑模板”窗口。单击“智能提示”→“显示”下拉列表框，单击“Column[2]-最低工资”选项，则显示“最低工资”列的信息（见图 7-6）。拖动 RangeValidator 控件到 EditItemTemplate 文本框右边。单击 RangeValidator1 控件，单击 ControlToValidate 属性并选择 TextBox1，单击 ForeColor 属性并选择 Red，单击 ErrorMessage 属性并输入 *，单击 MaxiumValue 属性并输入 100，单击 MiniumValue 属性并输入 1，单击 Type 属性并选择 Integer。以同样的方式给“Column[3]-最高工资”选项添加输入数据范围的检查功能。

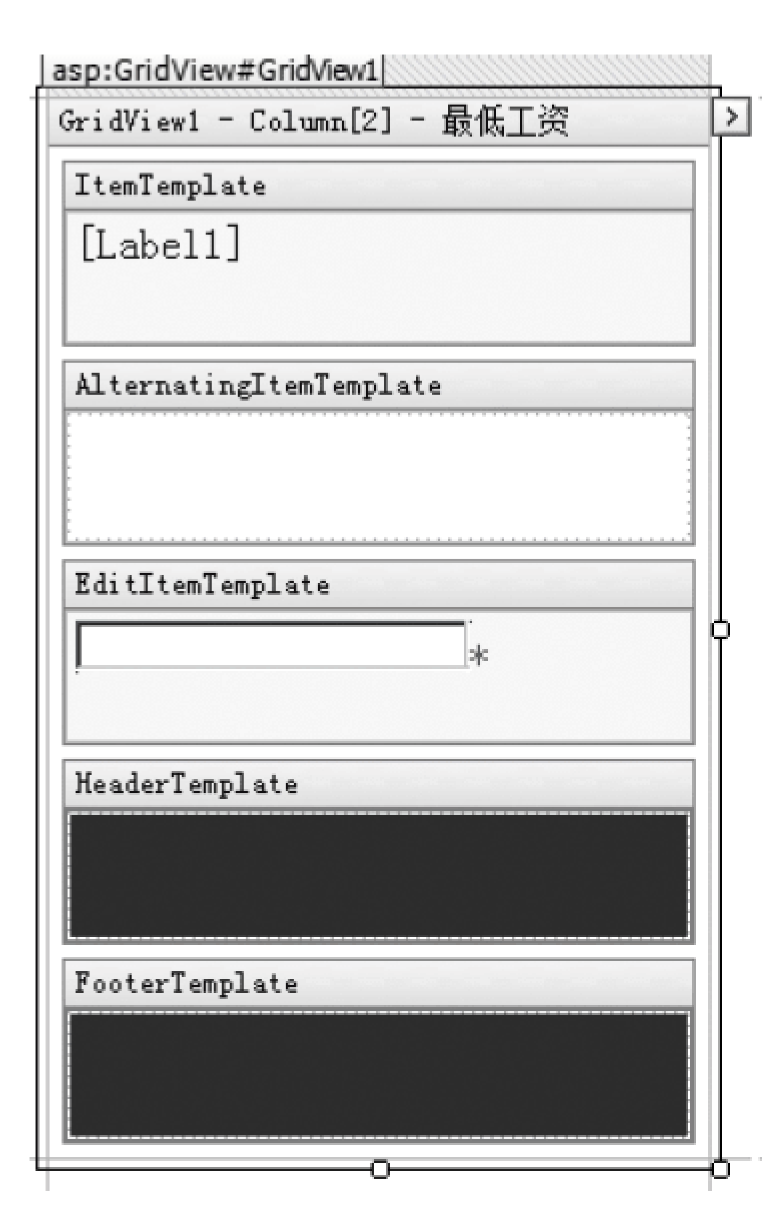


图 7-6　编辑模板

⑥ 单击 GridView1 控件，再单击“事件”按钮，然后双击 RowEditing 事件，进入代码编辑器并输入如下代码：

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
    GridView1.EditIndex=e.NewEditIndex;
    Bind();
}
```

⑦ 单击 GridView1 控件，再单击“事件”按钮，然后双击 RowCancelingEdit 事件，进入代码编辑状态，完成取消功能代码的编写。

```
protected void GridView1_RowCancelingEdit(object sender, GridViewCancelEdit-
EventArgs e)
{
    GridView1.EditIndex=-1;
    Bind();
}
```

⑧ 单击 GridView1 控件，再单击“事件”按钮，然后双击 RowUpdating 事件，进入代码编辑器，完成数据更新功能代码的编写。初始化最高工资为 10、最低工资为 10，现分别更新为 99 和 11（见图 7-7）。

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    String jid=GridView1.DataKeys[e.RowIndex][0].ToString();
    SqlConnection conn;
    SqlCommand cmd;
    String minv= (GridView1.Rows[e.RowIndex].Cells[2].FindControl("TextBox1")
        as TextBox).Text;
    String maxv= (GridView1.Rows[e.RowIndex].Cells[3].FindControl("TextBox2")
        as TextBox).Text;
    String sql=String.Format("update jobs set min_lvl={0}, max_lvl={1}
        where job_id={2} ",
        minv, maxv, jid);
    conn=new SqlConnection("Server=localhost; Database=Pubs;uid=sa;
        pwd=zzb");
    cmd=new SqlCommand(sql, conn);
    conn.Open();
    cmd.ExecuteNonQuery();
    conn.Close();
    GridView1.EditIndex=-1;
    Bind();
}
```

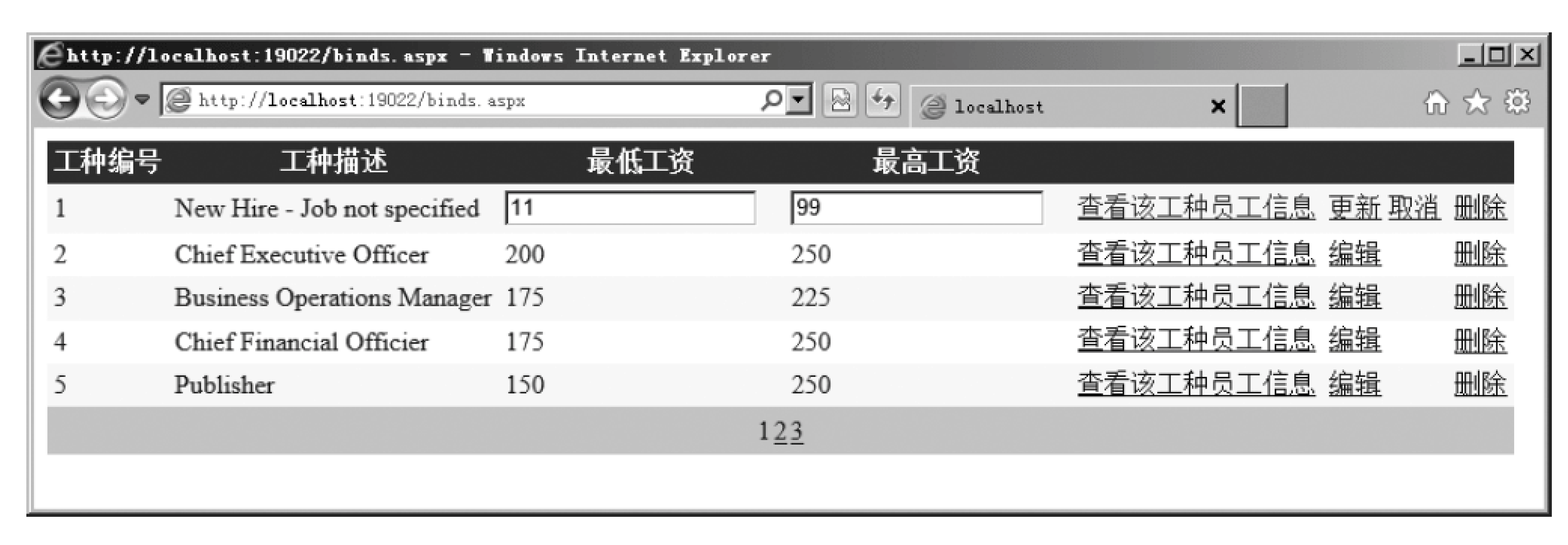


图 7-7　更新最高工资和最低工资

7.4　项 目 实 施

7.4.1　任务 1：制作管理员界面母版页

1. 任务目标

(1) 能熟练使用 CSS+DIV 制作页面。

(2) 能熟练使用 TreeView 制作树形导航。

2. 任务内容

（1）制作 AdminMP. master 母版页。

（2）完成管理母版页中的树形导航。

3. 任务实施步骤

该任务要求实现管理员所有操作界面的公共部分，即 AdminMP. master 母版页，并在母版页中使用 Tree 实现树形导航功能，如图 7-8 所示。



图 7-8　AdminMP. master 母版页

（1）创建 AdminMP. master 母版页。为结构清晰起见，右击网站根目录文件夹，单击“添加”→“新建文件夹”命令，输入 Admin。右击 Admin 文件夹，单击“添加”→“添加新项”命令，打开“添加新项”对话框，单击“母版页”选项，在“名称”文本框中输入 AdminMP . master。

（2）编写 Admin. css 样式表。右击网站根目录文件夹，单击“添加”→“新建文件夹”命令，输入 css。右击 css 文件夹，单击“添加”→“样式表”命令，输入 Admin. css。双击 Admin. css 文件，输入 css 样式代码。该样式代码的主要功能是设置 5 个 DIV 层的样式和位置布局。

```
body
{
    font-size:small;
}
#containerdiv
{
    width:800px; padding:0; margin:0 auto; height:100%; background-color:white;
}
#headerdiv
{
    width:800px; padding:0; margin:0 auto; height:180px;
    background-attachment: scroll * /
    background-image:url(http://localhost:19512/WorkSpace/Images/banner.png);
```

```
    background-repeat:no-repeat;
}
#navdiv
{
    width:20%;
    height:600px;
    float:left;
    overflow:hidden;
}
#contentdiv
{
    float:left;
    width:80%;
    height:600px;

}
#footdiv
{
    text-align:center;
}
```

（3）双击 AdminMP.master 文件，单击“源”按钮，切换到 HTML 页面。拖动 DIV 控件到<form>标记下，更改 id 为 containerdiv。拖动 DIV 控件到<div id="containerdiv">中，更改 id 为 headerdiv。拖动 DIV 控件到<div id="containerdiv">中，更改 id 为 navdiv。拖动 DIV 控件到<div id="containerdiv">中，更改 id 为 contentdiv，将<asp:ContentPlaceHolder>节剪切到<div id="contentdiv"> 中。拖动 DIV 控件到<div id="containerdiv">，更改 id 为 footdiv。详细的 HTML 代码如下。

```
<%@ Master Language="C#" AutoEventWireup="true"
     CodeFile="AdminMP.master.cs" Inherits="Admin_AdminMP" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
            "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="../css/Admin.css" rel="Stylesheet" />
</head>
<body>
    <form id="form1" runat="server">
    <div id="containerdiv">
        <div id="headerdiv">
        </div>
        <div id="navdiv">
        </div>
        <div id="contentdiv">
            <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">
                放置内容
            </asp:ContentPlaceHolder>
        </div>
```

```
            <div id="footdiv">
                CopyRight: NBCC
            </div>
        </div>
        </form>
    </body>
    </html>
```

(4) 拖动 TreeView 控件到 navdiv 层。单击智能提示,单击"编辑节点"菜单项,打开"节点编辑器"对话框,再单击"添加根节点"按钮,单击 Text 属性,输入"添加一级目录"。单击"添加根节点"按钮,再单击 Text 属性,输入"编辑一级目录"。单击"添加根节点"按钮,再单击 Text 属性,输入"添加二级目录"。单击"添加根节点"按钮,再单击 Text 属性,输入"编辑二级目录"。单击"添加根节点"按钮,再单击 Text 属性,输入"添加三级目录"。单击"添加根节点"按钮,再单击 Text 属性,输入"编辑三级目录"。单击"添加根节点"按钮,再单击 Text 属性,输入"添加商品"。单击"添加根节点"按钮,再单击 Text 属性,输入"编辑商品",如图 7-9 所示,最后单击"确定"按钮。单击 NodeStyle→ImageUrl,选择～/WebIcon/Green Ball. png 文件。



图 7-9　TreeView 节点编辑器

7.4.2　任务 2:一级目录的添加和编辑

1. 任务目标

(1) 能熟练使用 ADO. NET 技术插入数据。

(2) 能熟练使用 RequriedFieldValidator 验证非空数据。

(3) 能使用 GridView 显示、编辑和删除数据。

2. 任务内容

(1) 创建 InsertLevel1. aspx 页面。

(2) 实现向 T_Level1 中插入记录的功能。

(3) 实现一级目录的编辑和删除。

3. 任务实施步骤

(1) 设计 InsertLevel1.aspx 页面,如图 7-10 所示。创建 InsertLevel1.aspx 内容页,选择 AdminMP.master 母版页。拖动 TextBox 控件到"A 级目录名称"右侧。单击属性 ID,输入 TxtLevel1Name。拖动 TextBox 控件到"A 级目录描述"右侧,单击属性 ID,输入 TxtLevel1Desc。单击 TextMode 属性,选择 MultiLine 选项。拖动 RequiredFieldValidator 控件到 TxtLevel1Name 文本框右侧,单击 ControlToValidator 属性,选择 TxtLevel1Name 选项。单击 ErrorMessage 属性,输入"*"。拖动 Button 控件到窗口底部,单击 Text 属性,输入"添加"。单击 ID 属性,输入 BtnInsert。

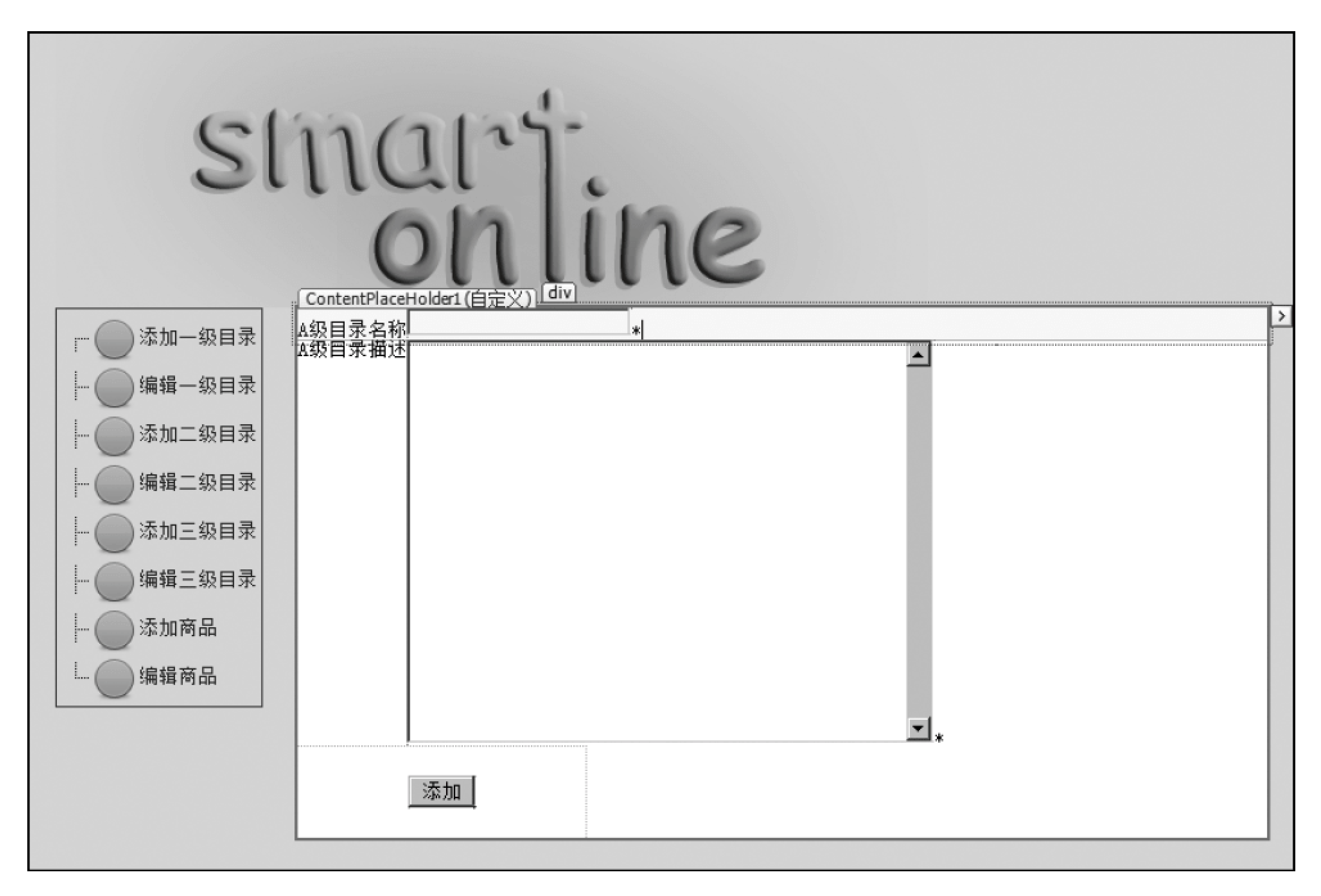


图 7-10 任务 2 布局

(2) 双击"添加"按钮。进入事件代码编辑状态。使用 using System.Data.SqlClient 指令引入 SqlClient 命名空间。在 BtnInsert_Click 事件中输入下列代码,实现插入功能。本书中多次使用 ADO.NET 技术访问数据库,所以该内容在本项目中没有作为必备知识再次讲解,读者若有疑惑,可参考项目 4。按 F5 功能键执行程序,效果见图 7-11。

```
protected void BtnInsert_Click(object sender, EventArgs e)
{
    String connStr=System.Configuration.ConfigurationManager.
        ConnectionStrings["SmartConnectionString"].ToString();
    SqlConnection con=new SqlConnection(connStr);
    con.Open();
    SqlCommand cmd=new SqlCommand();
    cmd.Connection=con;
    cmd.CommandText="Insert into T_Level1 values('"+
        TxtLevel1Name.Text+"','"+TxtLevel1Desc.Text+"')";
    cmd.ExecuteNonQuery();
    cmd.Dispose();
```

```
        con.Close();
    }
```

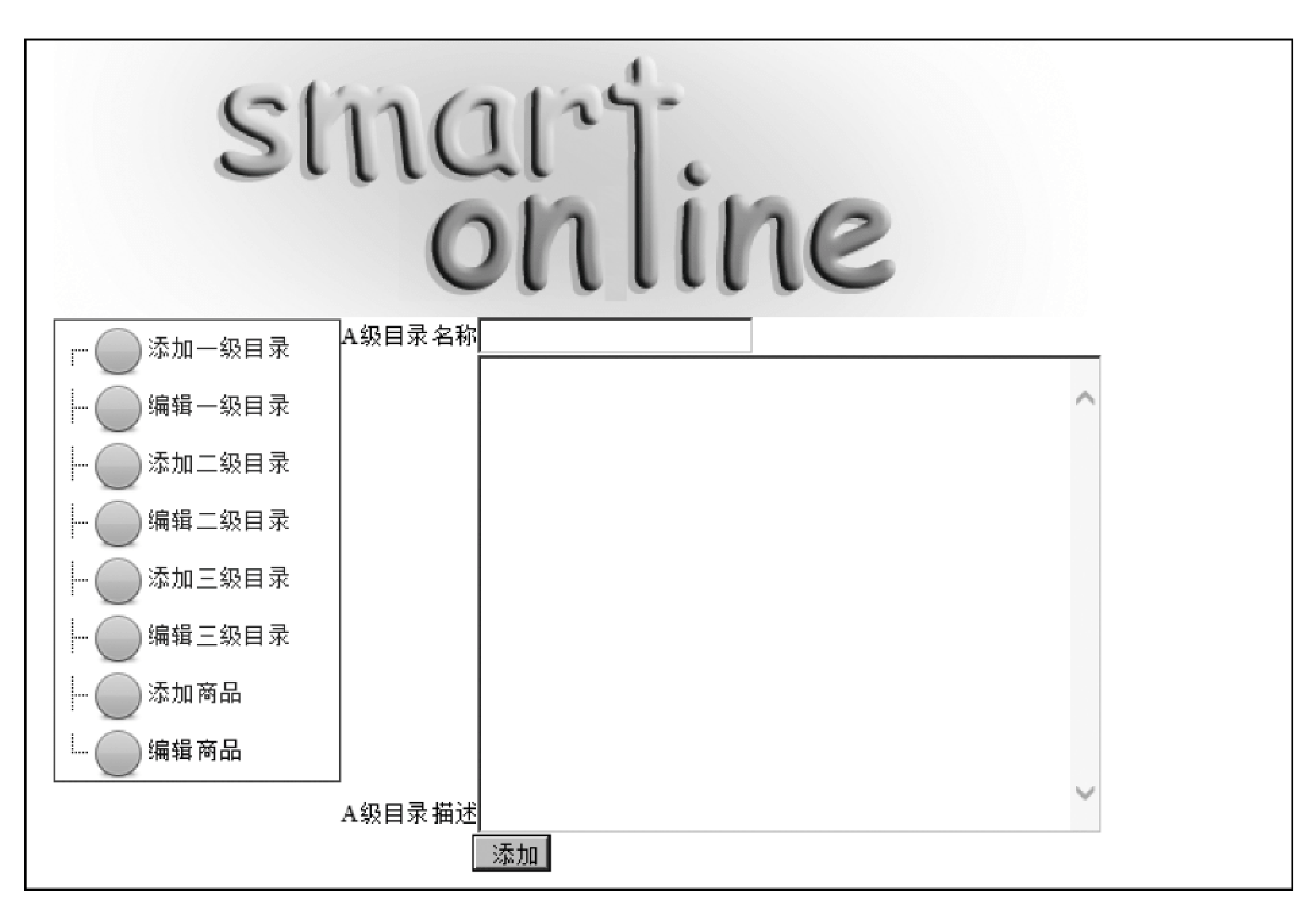


图 7-11　任务 2 中的插入一级目录的效果图

(3) 新建 EditLevel1.aspx 内容页，选择 AdminMP.master 母版页，拖动 GridView 控件到页面中，单击 ID 属性，输入 gvLevel1。按 F7 功能键，进入代码编辑页面，编写 Bind 方法来实现数据的绑定，代码如下。

```
    public void Bind()
    {
        SqlConnection con=new SqlConnection(System.Configuration.ConfigurationManager.
            ConnectionStrings["SmartConnectionString"].ToString());
        con.Open();
        SqlCommand cmd=new SqlCommand();
        cmd.Connection=con;
        cmd.CommandText="select * from T_Level1";
        SqlDataReader sdr=cmd.ExecuteReader();
        gvLevel1.DataSource=sdr;
        gvLevel1.DataBind();
    }
```

当页面第一次装载的时候，在 gvLevel1 中显示数据，所以需要在 Page_Load 事件中调用 Bind 方法，代码如下。

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if(!Page.IsPostBack)
            Bind();
    }
```

(4) 打开 gvLevel1"编辑列"窗口，单击 BoundField 选项，单击"添加"按钮，再单击

DataField 属性并输入 Level1_ID，单击 HeaderText 属性并输入“序号”，单击 Readonly 属性并设置为 true。单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 Level1_Name，单击 HeaderText 属性并输入“名称”，单击 Readonly 属性并设置为 true。单击 TemplateField 选项，单击“添加”按钮，单击 HeaderText 属性并输入“备注”。单击 CommandField 选项，单击“添加”按钮，再单击“确定”按钮，完成编辑列的操作。单击 gvLevel1 智能提示，单击“编辑模板”菜单项，打开“编辑模板”窗口，单击智能提示，再单击 Column[2]→EditItemTemplate；单击 TextBox1 控件，单击 ID 属性并输入 TxtLevel1Desc。拖动 RequiredFieldValidator 控件到 TxtLevel1Desc 文本框右侧，单击 ControlToValidate 属性并选择 TxtLevel1Desc，单击 ErrorMessage 属性并输入“ * ”。

(5) 单击 gvLevel1 控件，单击“事件”按钮，再双击 RowEditing 事件，输入下面的代码，实现编辑功能。

```
protected void GvLevel1_RowEditing(object sender, GridViewEditEventArgs e)
{
    GvLevel1.EditIndex=e.NewEditIndex;
    Bind();
}
```

(6) 单击 gvLevel1 控件，单击“事件”按钮，双击 RowCancelingEdit 事件，输入下面的代码，完成取消功能。

```
protected void GvLevel1_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
{
    GvLevel1.EditIndex=-1;
    Bind();
}
```

(7) 单击 gvLevel1 控件，单击“事件”按钮，双击 RowUpdating 事件，输入下面的代码，完成更新功能。在 RowUpdating 事件中首先通过 Cells[1].FindControl("TxtLevel3Name")方式查找模板列中的文本框并获取相应的 Text 值。接着采用 ADO.NET 方法调用 SQL 脚本并执行更新任务。

```
protected void GvLevel1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    String xh=  GvLevel1.Rows[e.RowIndex].Cells[0].Text;
    TextBox TxtLevel3Name= (TextBox)GvLevel1.Rows[e.RowIndex].
        Cells[1].FindControl("TxtLevel3Name");
    TextBox TxtLevel3Desc= (TextBox)GvLevel1.Rows[e.RowIndex].
        Cells[2].FindControl("TxtLevel3Desc");
    String Strcmd=String.Format("update T_level3 set Level3_Name='{0}',
        Level3_Desc='{1}' where Level3_ID={2}", TxtLevel3Name.Text,
        TxtLevel3Desc.Text, xh);
    try
    {
        sh.ExeNonQuery(Strcmd);
    }
    catch(Exception ex)
```

```
        {
            throw;
        }
        GvLevel1.EditIndex=-1;
        Bind();
    }
```

（8）单击 gvLevel1 控件，单击“事件”按钮，双击 RowDeleting 事件，输入下面的代码，实现删除的功能。该段代码没有使用 gvLevel1 的 DataKeyNames 属性，而是使用了第一列的值。如果需要将第一列的 ID 隐藏，那么就需要设置 DataKeyNames，然后在 RowDeleting 事件中获取绑定的 DataKeys 属性作为删除数据的条件。按 F5 功能键运行程序，效果如图 7-12 所示。

```
    protected void gvLevel1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        String seq=gvLevel1.Rows[e.RowIndex].Cells[0].Text;
        SqlConnection con=new SqlConnection(System.Configuration.ConfigurationManager.
        ConnectionStrings["SmartConnectionString"].ToString());
        con.Open();
        SqlCommand cmd=new SqlCommand();
        cmd.Connection=con;
        cmd.CommandText=String.Format("DELETE FROM T_Level1
            where Level1_ID={0}", seq);
        cmd.ExecuteNonQuery();
        cmd.Dispose();
        con.Close();
        Bind();
    }
```



图 7-12　任务 2 中的编辑和删除一级目录的效果图

7.4.3　任务 3：二级目录的添加和编辑

1. 任务目标

（1）能熟练使用 GridView 编辑和删除数据。

（2）能熟练使用 ADO. NET 技术执行数据库的各种操作。

2. 任务内容

（1）实现二级目录添加的功能。

（2）实现二级目录更新的功能。

（3）实现二级目录删除的功能。

3. 任务实施步骤

该任务的实现方法与任务 2 非常类似。

（1）根据图 7-13 设计 InsertLevel2. aspx 页面。新建 InsertLevel2. aspx 内容页，选择 AdminMP. master 母版页。现在读者应该熟悉添加控件的方法了，所以本书后面部分就不再讲述添加控件的方法了。

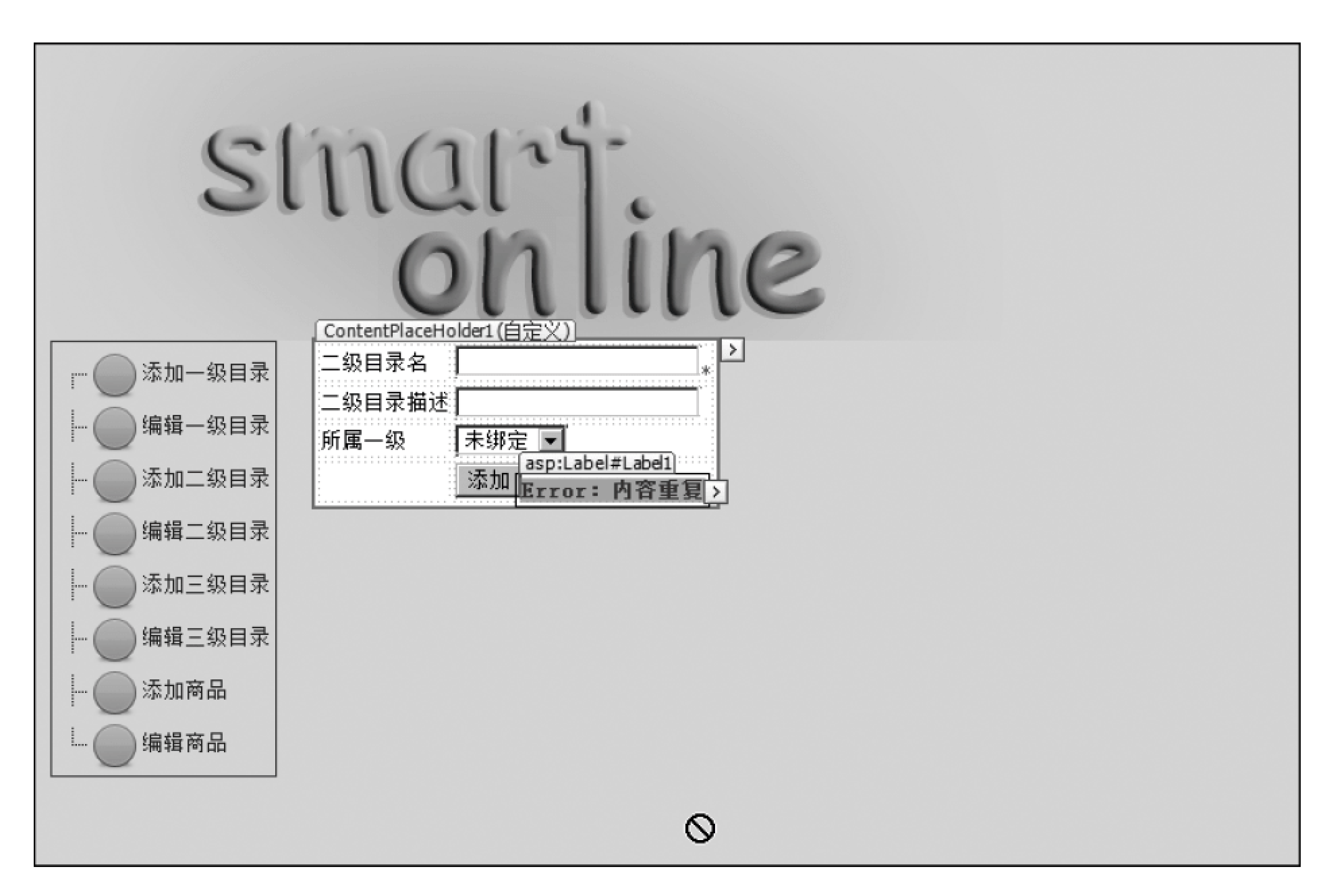


图 7-13　InsertLevel2. aspx 页面的布局

（2）当页面第一次装载时需要绑定和显示“所属一级”下拉列表的值。到目前为止，读者已经多次编写过 ADO. NET 访问数据的代码段了，这里不再赘述。以前项目中编写过 SqlHelper 类，现在就再次使用它。下面的代码展示了 SqlHelper 类的强大。开发人员只需要使用 SqlHelper sh＝new SqlHelper( ) 生成 sh 对象，再直接调用 sh 对象的 QueryOperation 查询方法即可。

```
SqlHelper sh=new SqlHelper();
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {
        DropDownList1.DataSource=sh.QueryOperation("select Level1_ID,
            Level1_Name from T_Level1");
        DropDownList1.DataBind();
    }
}
```

（3）双击“添加”按钮，进入“添加”按钮的 Click 事件的代码区，输入下列的代码：

```
try
{
    String StrCmd= String.Format("Insert into T_Level2 values('{0}','{1}',{2})",
        TextBox1.Text,
        TextBox2.Text, DropDownList1.SelectedValue);
    if(sdr != null)
        sdr.Close();
    sh.ExeNonQuery(StrCmd);
    con.Close();
}
catch(Exception ex)
{
    Label1.Visible= true;
}
```

（4）新建 EditLevel2.aspx 内容页，选择 AdminMP.master 母版页。拖动 GridView 控件到页面中。在“编辑列”窗口中单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 level2_id，单击 HeaderText 属性并输入序号，单击 Readonly 属性并设置为 true。在“编辑列”窗口中单击 TemplateField 选项，单击“添加”按钮，单击 HeaderText 属性并输入“二级目录名”。在“编辑列”窗口中单击 TemplateField 选项，单击“添加”按钮，单击 HeaderText 属性并输入“二级目录描述”。在“编辑列”窗口中单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 level1_name，单击 HeaderText 属性并输入“一级目录”，单击 Readonly 属性并设置为 true。在“编辑列”窗口中单击 CommandField 选项，单击“添加”按钮。单击“确定”按钮完成编辑列的操作。

（5）在 GridView1 编辑模板窗口中，对 Column[1]的 EditItemTemplate 文本框进行必填验证功能的实现，如图 7-14 所示。单击 TextBox 控件，单击 ID 属性并输入 TxtLevel2Name。拖动 RequiredFieldValidator 到 TxtLevel2Name 文本框右侧，单击

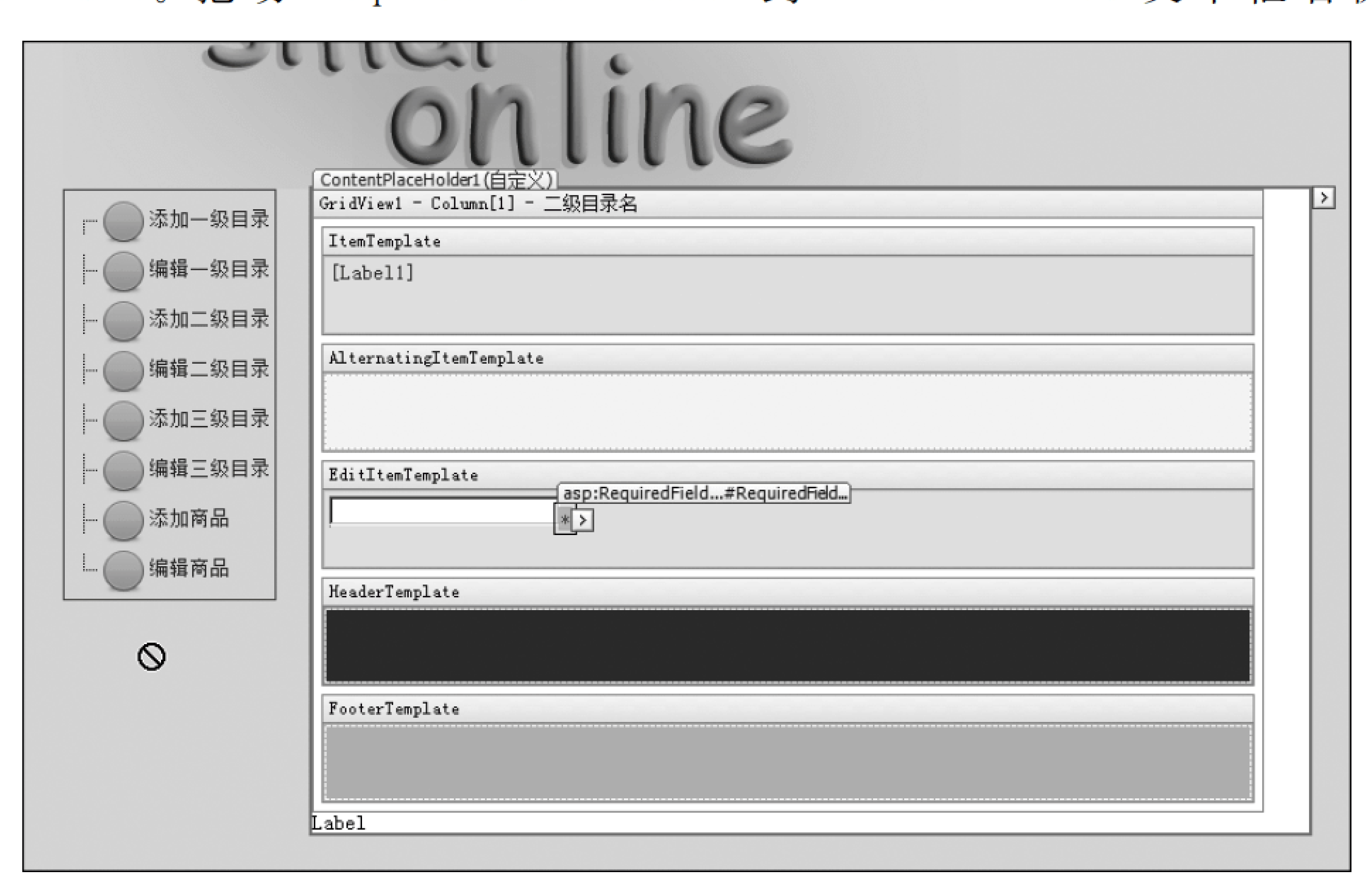


图 7-14　二级目录名模板列

ControlToValidator 属性并选择 TxtLevel2Name；单击 ErrorMessage 属性输入 * 。单击 GridView1 智能提示，单击"显示"下拉列表中的 Column[2]。单击 TextBox 控件，单击 ID 属性并输入 TxtLevel2Desc。拖动 RequiredFieldValidator 到 TxtLevel2Desc 文本框右侧，单击 ControlToValidator 属性并选择 TxtLevel2Desc；单击 ErrorMessage 属性并输入"＊"。这样 GridView 的列绑定功能就实现了，效果如图 7-15 所示。



图 7-15　任务 3 中的编辑二级目录列绑定的功能

（6）任务 3 中对 GridView 的编辑、取消编辑、更新和删除操作跟任务 2 中十分类似，所以这个步骤就简单将代码展示一下。任务 3 的最终执行效果图如图 7-16～图 7-18 所示。

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
    GridView1.EditIndex=e.NewEditIndex;
    Bind();
}
protected void GridView1_RowCancelingEdit(object sender,
                                GridViewCancelEditEventArgs e)
{
    GridView1.EditIndex=-1;
    Bind();
}
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    String level2_id=GridView1.Rows[e.RowIndex].Cells[0].Text;
    TextBox txtname  = (TextBox)GridView1.Rows[e.RowIndex].
            Cells[1].FindControl("TxtLevel2Name");
    TextBox txtdesc  = (TextBox)GridView1.Rows[e.RowIndex].
            Cells[2].FindControl("TxtLevel2Desc");
    String strCmd=String.Format(
        "update T_Level2 set Level2_Name='{0}',Level2_Desc='{1}' where
        Level2_ID={2}",
        txtname.Text, txtdesc.Text,level2_id);
    sh.ExeNonQuery(strCmd);
    GridView1.EditIndex=-1;
    Bind();
}
```

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    String Level2_ID=GridView1.Rows[e.RowIndex].Cells[0].Text;
    String StrCmd=String.Format("delete from T_level2 where Level2_ID={0}",
        Level2_ID);
    if(!sh.ExeNonQuery(StrCmd))
    {
        Label3.Text="ERRRR";
    }
    GridView1.EditIndex=-1;
    Bind();
}
```

smart online

添加一级目录
编辑一级目录
添加二级目录
编辑二级目录
添加三级目录
编辑三级目录
添加商品
编辑商品

序号	二级目录名	二级目录描述	一级目录		
27	电子书	电子书	图书、音像	编辑	删除
28	数字音乐	数字音乐	图书、音像	编辑	删除
29	大家电	大家电	家用电器	编辑	删除
30	厨房电器	厨房电器	家用电器	编辑	删除
31	电脑整机	电脑整机	电脑	编辑	删除
32	电脑配件	电脑配件	电脑	编辑	删除
33	男装	男装	服饰	编辑	删除
34	女装	女装	服饰	编辑	删除

图 7-16　任务 3 的编辑界面



图 7-17　任务 3 的更新界面



图 7-18　任务 3 的删除界面

7.4.4　任务 4：三级目录的添加和编辑

1. 任务目标

(1) 能熟练使用 GridView 编辑和删除数据。

(2) 能熟练使用 ADO. NET 技术执行数据库操作。

2. 任务内容

(1) 实现三级目录添加的功能。

(2) 实现三级目录更新的功能。

(3) 实现三级目录删除的功能。

3. 任务实施步骤

该任务实现的具体操作跟任务 3 十分相似。作者将这个任务安排到这个项目中只是为了任务的完整性,所需的知识几乎跟任务 3 一模一样。

(1) 根据图 7-19 设计 InsertLevel3. aspx 页面。新建 InsertLevel3. aspx 内容页,选择 AdminMP. master 母版页。

(2) 登录 SQL Server 2008 服务器,单击"新建查询"按钮,在"查询分析器"中创建 v_level1_b2 视图。

```
CREATE VIEW v_level1_b2
AS
SELECT    T_Level2.Level2_ID, T_Level2.Level1_ID, T_Level1.Level1_Name
FROM      T_Level2 INNER JOIN
          T_Level1 ON T_Level2.Level1_ID=T_Level1.Level1_ID
```

当页面第一次装载时,需要绑定和显示"一级目录"和"二级目录"的列表值,在 Page_Load 事件中输入下列代码:

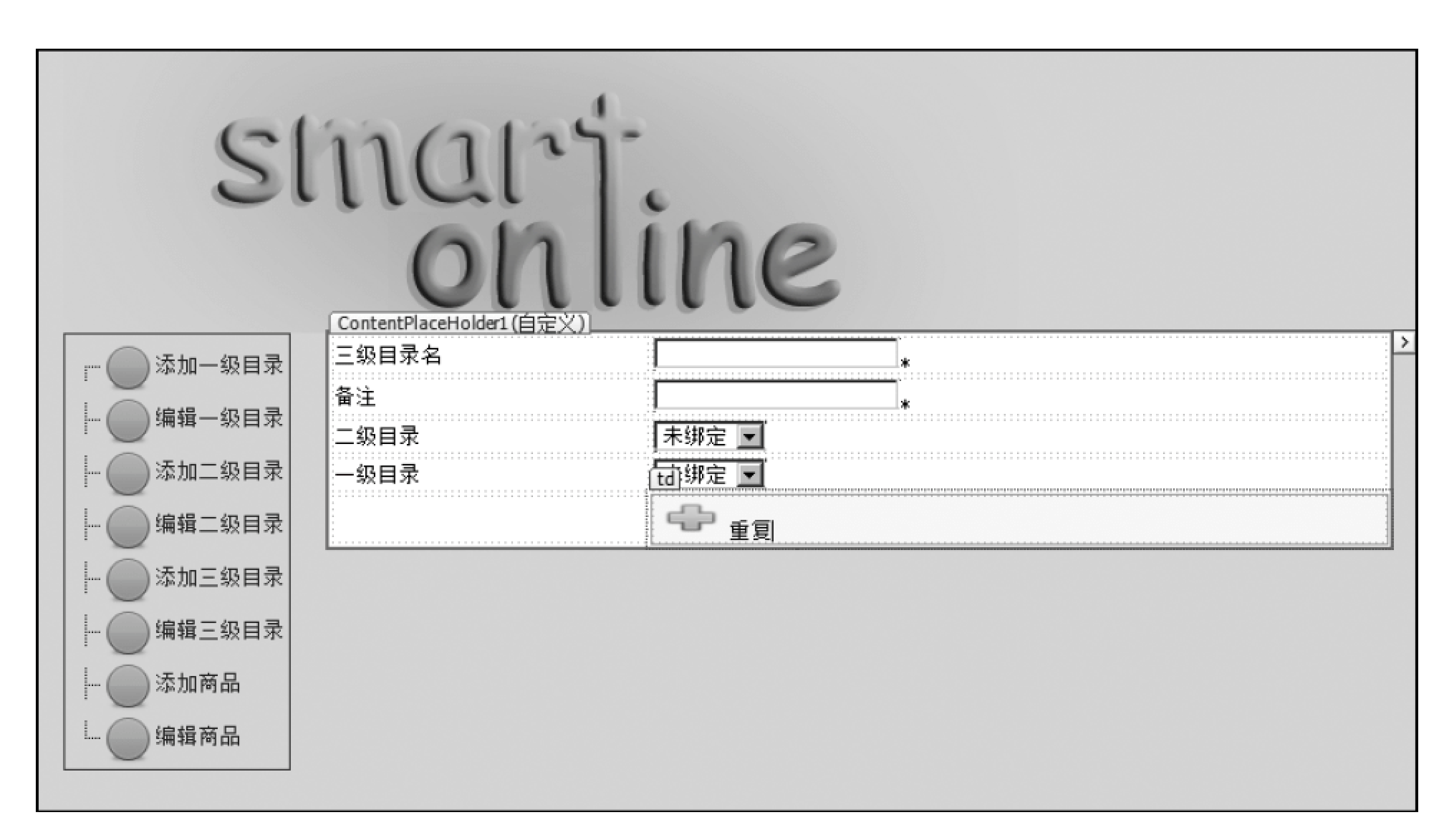


图 7-19 InsertLevel3.aspx 页面的布局

```
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {
        DropDownList1.DataSource= sh.QueryOperation("select * from T_Level2");
        DropDownList1.DataBind();
        DropDownList2.DataSource= sh.QueryOperation(String.Format("select * from
            v_level1_b2 where level2_id= {0}",
            DropDownList1.SelectedValue));
        DropDownList2.DataBind();
    }
}
```

(3) 双击 DropDownList1 控件，进入 DropDownList1_SelectedIndexChanged 事件的代码编辑区，输入下列代码，实现二级目录改变时也会动态改变一级目录名的效果。

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    DropDownList2.DataSource= sh.QueryOperation(String.Format("select * from
        v_level1_b2 where level2_id= {0}",
        DropDownList1.SelectedValue));
    DropDownList2.DataBind();
}
```

(4) 双击“添加”按钮，进入“添加”按钮的 Click 事件的代码编辑区，输入下列代码：

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    Label1.Visible= false;
    try
    {
        sh.ExeNonQuery(String.Format("insert into T_Level3 values('{0}','{1}',{2},{3})",
            TextBox1.Text,
```

```
                TextBox2.Text,
                DropDownList1.SelectedValue, DropDownList2.SelectedValue));
            sh.closeConn();
        }
        catch(Exception ex)
        {
            Label1.Visible=true;
        }
    }
```

(5) 新建 EditLevel3.aspx 内容页,选择 AdminMP.master 母版页。拖动 GridView 控件到页面中。在"编辑列"窗口中单击 BoundField 选项,单击"添加"按钮,单击 DataField 属性并输入 level3_id。单击 HeaderText 属性输入序号,单击 Readonly 属性并设置为 true。在"编辑列"窗口中单击 TemplateField 选项,单击"添加"按钮,单击 HeaderText 属性并输入"三级目录名"。在"编辑列"窗口中单击 TemplateField 选项,单击"添加"按钮,单击 HeaderText 属性并输入"备注"。在"编辑列"窗口中单击 BoundField 选项,单击"添加"按钮,单击 DataField 属性并输入 level1_name,单击 HeaderText 属性并输入"一级目录",单击 Readonly 属性并设置为 true。在"编辑列"窗口中单击 BoundField 选项,单击"添加"按钮,单击 DataField 属性并输入 level2_name,单击 HeaderText 属性并输入"二级目录",单击 Readonly 属性并设置为 true。在"编辑列"窗口中单击 CommandField 选项,单击"添加"按钮。单击"确定"按钮完成编辑列的操作。

(6) 在 GridView1 编辑模板窗口中,对 Column[1]的 EditItemTemplate 文本框进行内容的必填验证。单击 TextBox 控件,单击 ID 属性并输入 Txtlevel3Name。拖动 RequiredFieldValidator 到 Txtlevel3Name 文本框右侧,单击 ControlToValidator 属性并选择 Txtlevel3Name;单击 ErrorMessage 属性并输入"*"。单击 GridView1 智能提示,单击"显示"下拉列表中的 Column[2]。单击 TextBox 控件,单击 ID 属性并输入 TxtLevel3Desc。拖动 RequiredFieldValidator 到 TxtLevel3Desc 文本框右侧,单击 ControlToValidator 属性并选择 TxtLevel3Desc;单击 ErrorMessage 属性并输入"*"。这样 GridView 的列绑定操作就完成了。

(7) 任务 4 中对 GridView 的编辑、取消编辑、更新和删除操作跟任务 3 非常类似,所以这个步骤只简单将代码展示一下。任务 4 的最终执行效果图如图 7-20 和图 7-21 所示。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Admin_Level3Management_EditLevel3 : System.Web.UI.Page
{
    SqlHelper sh=new SqlHelper();
    protected void Page_Load(object sender, EventArgs e)
    {
```

```
            if(!Page.IsPostBack)
            {
                Bind();
            }
        }
        private void Bind()
        {
            GridView1.DataSource= sh.QueryOperation("select * from v_level3");
            GridView1.DataBind();
        }
        protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
        {
            GridView1.EditIndex= e.NewEditIndex;
            Bind();
        }
        protected void GridView1_RowCancelingEdit(object sender,
            GridViewCancelEditEventArgs e)
        {
            GridView1.EditIndex= - 1;
            Bind();
        }
        protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
        {
            String xh=  GridView1.Rows[e.RowIndex].Cells[0].Text;
            TextBox TxtLevel3Name= (TextBox)GridView1.Rows[e.RowIndex].Cells[1].
                FindControl("TxtLevel3Name");
            TextBox TxtLevel3Desc= (TextBox)GridView1.Rows[e.RowIndex].Cells[2].
                FindControl("TxtLevel3Desc");
            String Strcmd= String.Format("update T_level3 set Level3_Name= '{0}',
                Level3_Desc= '{1}' where Level3_ID= {2}", TxtLevel3Name.Text,
                TxtLevel3Desc.Text, xh);
            try
            {
                sh.ExeNonQuery(Strcmd);
            }
            catch(Exception ex)
            {

                throw;
            }
            GridView1.EditIndex= - 1;
            Bind();
        }
    }
```

7.4.5　任务 5：商品信息的添加

1. 任务目标

（1）能熟练使用 ADO. NET 访问数据库。

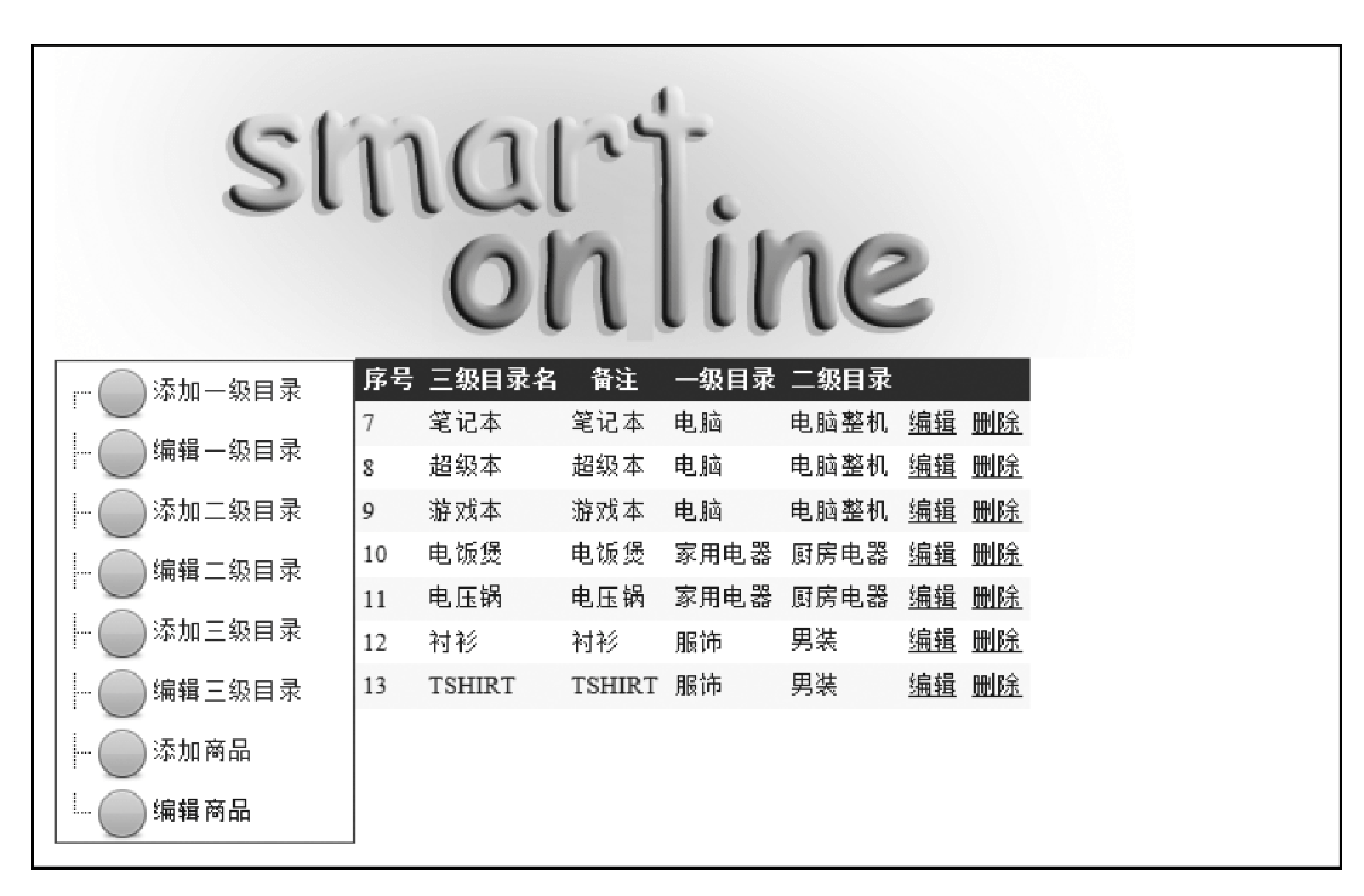


图 7-20　任务 4 中的编辑界面



图 7-21　任务 4 中的更新界面

(2) 能熟练使用 FileUpLoad 上传控件。

(3) 能使用 JavaScript 实现客户端界面的交互。

2. 任务内容

(1) 设计商品的添加页面。

(2) 实现动态添加商品额外信息表的功能。

(3) 实现商品样图的上传。

(4) 实现商品信息的保存。

3. 任务实施步骤

商品信息添加过程实施难度较大。商品信息分为商品基本信息和具体信息。其基本流

程是：首先输入商品的基本信息和所属的三级目录；接着选择商品具体类，若没有所属的商品具体类，则单击下拉按钮，显示商品具体类创建界面(见图 7-22)；在商品具体类创建界面中输入列名和类型(见图 7-23)，单击“保存”按钮，将数据插入数据库；若已经有商品具体类，则选择商品“具体类别”下拉列表，显示商品具体信息输入界面。接着单击“浏览”按钮选择图片，单击“上传”按钮上传图片到服务器(见图 7-24)，并将路径保存到数据库中。最后单击“添加”按钮保存商品信息到数据库中。

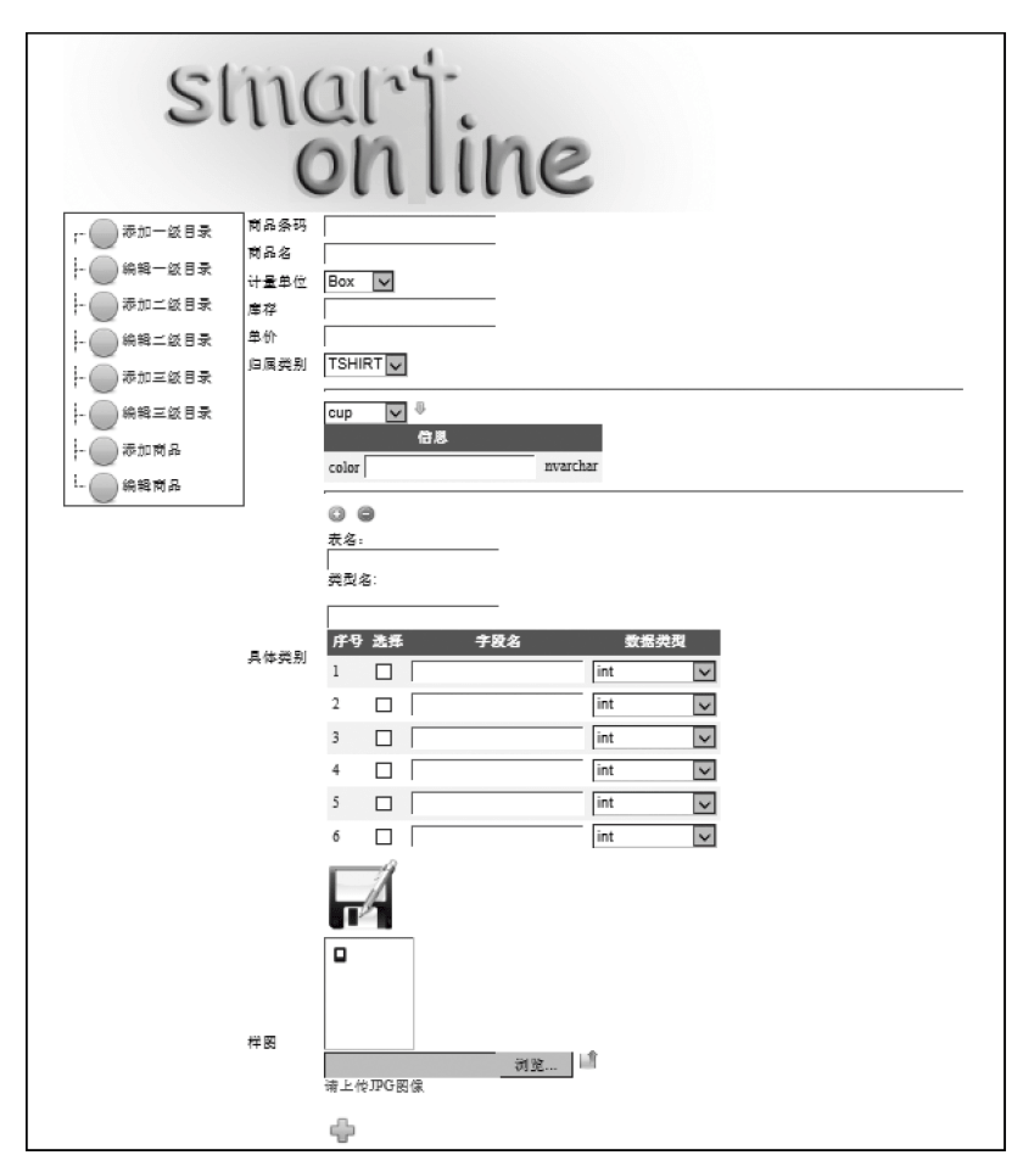


图 7-22　显示创建具体类别

① 新建 AddItem. aspx 内容页，选择 AdminMP. master 母版页。拖动 Html 标签页中的 Table 控件到页面中。按住 Ctrl+Alt+↓组合键插入一行，总共为 10 行。按住 Ctrl+Alt+→组合键插入一列，总共为 2 列。

② 拖动 TextBox 控件到第 1 行第 2 列。拖动 TextBox 控件到第 2 行第 2 列。拖动 DropDownList 控件到第 1 行第 2 列。单击智能提示，单击“编辑项”菜单项，单击“添加”按钮，单击 Text 属性并输入 Box；单击“添加”按钮，单击 Text 属性并输入 Group。拖动 TextBox 控件到第 4 行第 2 列。拖动 TextBox 控件到第 5 行第 2 列。拖动 DropDownList 控件到第 6 行第 2 列。拖动 DropDownList 控件到第 7 行第 2 列；拖动 GridView 控件到第 7 行第 2 列，单击 ID 属性并输入 GridView2。拖动 Panel 控件到第 8 行第 2 列，ID 为 Panel1。拖动 ImageButton 控件到 Panel1，单击 ImageUrl 属性并选择 ～/WebIcon/

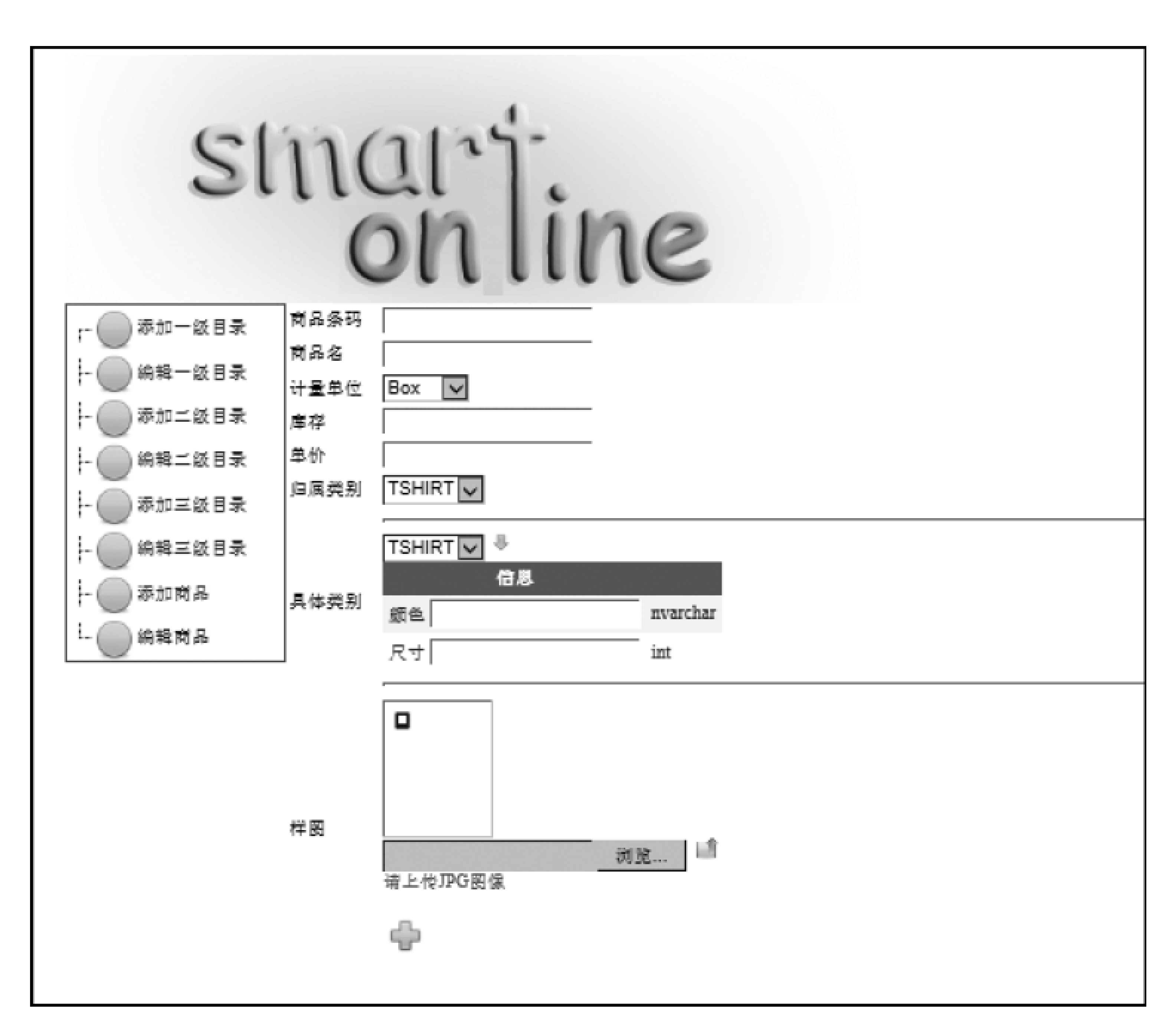


图 7-23　显示商品具体信息输入界面

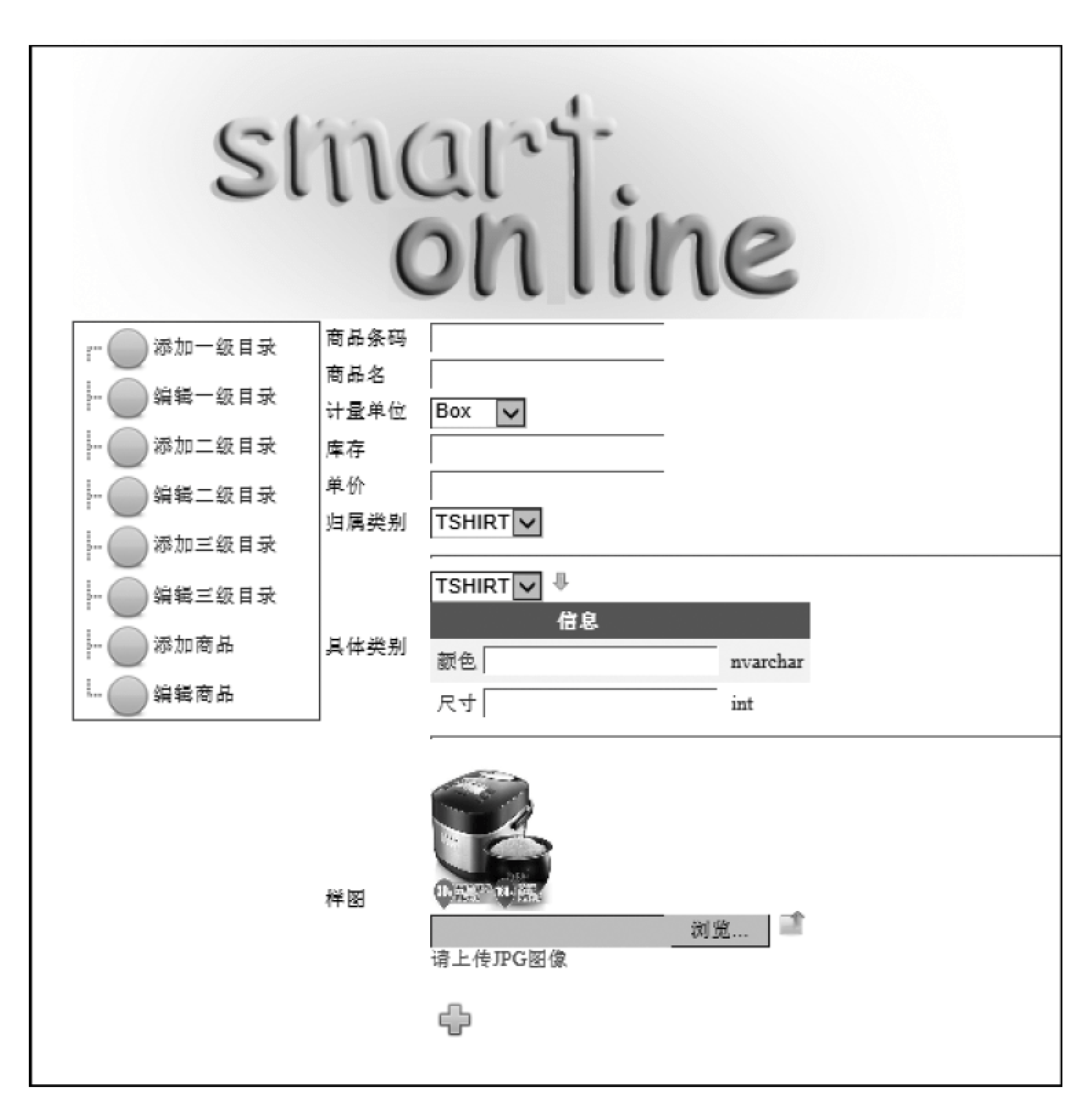


图 7-24　上传商品样图

add-s.png 图片文件；拖动 ImageButton 控件到 Panel1，单击 ImageUrl 属性并选择 ~/WebIcon/delete-s.png 图片文件；在 Panel1 的 ImageButton2 后按 Enter 键并输入“表名”，拖动 TextBox 控件到“表名”文字下方，单击 ID 属性并输入 TxtTableName。在

TxtTableName 文本框后按 Enter 键并输入“类型名”，拖动 TextBox 控件到“类型名”文字下方，单击 ID 属性并输入 TxtTypeName。拖动 GridView 控件到 TxtTypeName 文本框下，单击 ID 属性并输入 GridView1。拖动 ImageButton 控件到 GridView1 控件下，单击 ID 属性并输入 BtnSaveInfo。拖动 Image 控件到第 9 行第 2 列。拖动 FileUpLoad 控件到 Image1 控件下，拖动 ImageButton 控件到 FileUpLoad1 右侧，单击 ImageUrl 属性并选择～/WebIcon/Send Document. png 图片文件。拖动 ImageButton 控件到第 10 行第 2 列，单击 ImageUrl 属性并选择～/WebIcon/Add. png 图片文件。

③ 绑定数据到“归属类别”下拉列表框。在页面第一次装载的时候需要显示商品所属的三级目录，所以需要在 Page_Load 中绑定“归属类别”下拉列表框，然后输入下列代码。

```
SqlHelper sh=new SqlHelper();
public void BindLevel3()
{
    DropDownList2.DataSource=sh.QueryOperation("select Level3_ID,
        Level3_Name from T_Level3");
    DropDownList2.DataBind();
}
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {

        BindLevel3();
    }
}
```

④ 绑定数据到“具体类别”下拉列表框。在页面第一次装载的时候需要显示所有的具体类别以供管理员选择，所以需要在 Page_Load 中绑定“具体类别”下拉列表框，输入下列代码。

```
public void BindLX()
{
    ddlJDLX.DataSource=sh.QueryOperation("select DTName, DTTableName
        from T_DetailsType");
    ddlJDLX.DataBind();
}
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {

        BindLevel3();
        BindLX
    }
}
```

⑤ 当用户选择“具体类别”下拉列表中的值时，GridView2 显示所选的具体类别所对应的列信息。编辑 GridView2 列绑定。单击 TemplateField 选项，单击“添加”按钮，单击 HeaderText 属性并输入“信息”。单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 Data_Type。双击“具体类别”下拉列表，进入 SelectedIndexChanged 事件代码编写区，输入下列代码。

```
public void DisplayColumnInfo2()
{
    String sql2=String.Format("select column_name, Data_Type from
        information_schema.columns where Table_Name='{0}'
        and column_name not in('IID','DTID')",
        ddlJDLX.SelectedValue.ToString());
    GridView2.DataSource=sh.QueryOperation(sql2);
    GridView2.DataBind();
}
protected void ddlJDLX_SelectedIndexChanged(object sender, EventArgs e)
{
    DisplayColumnInfo2();
}
```

⑥ 编辑 GridView1 列。GridView1 默认显示的是创建表的界面，包括字段名和字段类型。单击 GridView1 智能提示，单击“编辑列”菜单，打开“编辑列”窗口。单击 TemplateField 选项，单击“添加”按钮，单击 HeaderText 属性并输入“序号”。单击 TemplateField 选项，单击“添加”按钮，单击 HeaderText 属性并输入“选择”。单击 TemplateField 选项，单击“添加”按钮，单击 HeaderText 属性并输入“字段名”。单击 TemplateField 选项，单击“添加”按钮，单击 HeaderText 属性并输入“数据类型”。单击“源”按钮，切换到 HTML 代码页面，在“序号”列中修改成如下代码，其中“<% # Container. DataItemIndex+1 %>”代码的作用是实现顺序编号显示，如 1，2，…。

```
<asp:TemplateField HeaderText="序号">
    <ItemTemplate>
        <%#Container.DataItemIndex+1 %>
    </ItemTemplate>
    <HeaderStyle Wrap="False" />
</asp:TemplateField>
```

单击“设计”按钮，切换到设计视图，单击智能提示，单击“编辑模板”菜单项，打开“编辑模板”窗口。单击“编辑模板”菜单项，单击“显示”列表中的“Column[1]-选择”选项，拖动 CheckBox 到 ItemTemplate 中，单击 ID 属性并输入 chkID。单击“显示”列表中的“Column[2]-字段名”选项，拖动 TextBox 到 ItemTemplate 中，单击 ID 属性并输入 TxtColumnName。单击“显示”列表中的“Column[3]-数据类型”选项，拖动 DropDownList 到 ItemTemplate 中，单击 ID 属性并输入 DDLDataType，单击“编辑项”菜单项，单击“添加”按钮，单击 Text 属性并输入 int。单击“添加”按钮，单击 Text 属性并输入 float。单击“添加”按钮，单击 Text 属性并输入 nvarchar(500)。按 F7 功能键切换到代码页面，编写 BindTM 方法，在 Page_Load 中调用。代码如下。

```
public void BindIM()
{
    dt.Columns.Clear();
    //dt.Clear();
    dt.Columns.Add(new DataColumn("ColumnID"));
    dt.Columns.Add(new DataColumn("ColumnName"));
    dt.Columns.Add(new DataColumn("ColumnType"));
    DataRow dr= dt.NewRow();
    dr["ColumnID"]=Guid.NewGuid().ToString();
    dt.Rows.Add(dr);
    GridView1.DataSource= dt;
    GridView1.DataBind();
}
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {

        BindLevel3();
        PanelInfo.Visible= false;
        BindIM();
        BindLX();
        DisplayColumnInfo2();
    }
}
```

⑦ 实现“具体类别创建表列”的增加和删除功能。双击 ImageButton2 按钮，输入下列代码。

```
protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
{
    DataRow dr= dt.NewRow();
    dr["ColumnID"]=Guid.NewGuid().ToString();
    dt.Rows.Add(dr);
    GridView1.DataSource= dt;
    GridView1.DataBind();
}
```

双击 ImageButton3 按钮，输入下列代码。

```
protected void ImageButton3_Click(object sender, ImageClickEventArgs e)
{
    for(int i= 0; i < GridView1.Rows.Count; i++)
    {
        CheckBox ck= (CheckBox)GridView1.Rows[i].Cells[1].FindControl("chkID");
        String colID= GridView1.DataKeys[0].Value.ToString();
        if(ck.Checked)
        {
            foreach(DataRow dr in dt.Rows)
            {
                if(dr["ColumnID"].ToString()==colID)
```

```
                    {
                        dt.Rows.Remove(dr);
                        break;
                    }
                }
            }
        }
        GridView1.DataSource=dt;
        GridView1.DataBind();
    }
```

⑧ 实现创建具体类别表的功能。双击 BtnSaveInfo 按钮，进入代码页面并输入下列代码。

```
protected void BtnSaveInfo_Click(object sender, ImageClickEventArgs e)
{
    sh.ExeNonQuery(String.Format("Insert into T_DetailsType values('{0}','{1}','{2}')",
        TxtTypeName.Text,
        TxtTableName.Text, ""));
    //创建详细类型表
    String createTableStr="Create Table "+TxtTableName.Text+"( IID int
        primary key
        identity(1,1),";
    for(int i=0; i<GridView1.Rows.Count; i++)
    {
        TextBox tb=(TextBox)GridView1.Rows[i].Cells[2].
            FindControl("TxtColumnName");
        if(tb.Text !="")
        {
            DropDownList ddl=(DropDownList)GridView1.Rows[i].Cells[3].
                FindControl("DDLDataType");
            String StrType=tb.Text+" "+ddl.SelectedValue.ToString();
            StrType=StrType+",";
            createTableStr=createTableStr+StrType;
        }
    }
    createTableStr=createTableStr+"DTID int)";
    sh.ExeNonQuery(createTableStr);
    String fk=String.Format("alter table {0} add constraint {1} foreign
    key(DTID)
        references T_DetailsType(DTID)", TxtTableName.Text,
        "fk"+Guid.NewGuid().ToString().Substring(0,7));
    sh.ExeNonQuery(fk);
    BindLX();
    ImageButton1_Click(sender, e);
}
```

⑨ 实现上传功能。双击 ImageButton4 按钮并进入代码页面，输入下列代码。

```
protected void ImageButton4_Click(object sender, ImageClickEventArgs e)
{
```

```
        if(FileUpload1.HasFile)
        {
            String fileExt=FileUpload1.FileName.ToString();
            if(fileExt.Contains(".jpg"))
            {
                String phyPath=Server.MapPath("~/TP");
                FileUpload1.SaveAs(phyPath+"//"+fileExt);
                Image1.ImageUrl="~/TP/"+fileExt;
            }
            else
            {
                Label1.Visible=true;
            }
        }
    }
```

⑩ 实现保存商品信息功能。双击 ImageButton5 按钮,进入代码页面,输入下列代码。

```
protected void ImageButton5_Click(object sender, ImageClickEventArgs e)
{
    int DTID=-1;
    int IID=-1;
    SqlTransaction trans;
    SqlConnection con=sh.returnConn();
    con.Open();
    trans=con.BeginTransaction();
    try
    {
        SqlCommand cmd=new SqlCommand();
        cmd.Transaction=trans;
        cmd.Connection=con;
        String sql1=String.Format("select DTID from T_DetailsType where
          DTTableName='{0}'", ddlJDLX.SelectedValue.ToString());
        cmd.CommandText=sql1;
        SqlDataReader sdr=cmd.ExecuteReader();
        if(sdr.HasRows)
        {
            sdr.Read();
            DTID=sdr.GetInt32(0);
        }
        sdr.Close();
        String sql2=String.Format("insert into {0} values(",ddlJDLX
            .SelectedValue.ToString()));
        int count=GridView2.Rows.Count;
        for(int i=0; i<count; i++)
        {
            TextBox tb=(TextBox)GridView2.Rows[i].Cells[0].FindControl
                ("TxtValue");
            String kind=GridView2.Rows[i].Cells[1].Text;
            if(kind.Equals("nvarchar"))
            {
```

```
                sql2=sql2+"'"+tb.Text+"'";
            }else if(kind.Equals("int"))
            {
                sql2=sql2 +tb.Text;
            }
            sql2=sql2+",";
        }
        sql2=sql2+DTID+"); select cast(scope_identity() as int)";
        cmd.CommandText=sql2;
        IID=(int)cmd.ExecuteScalar();
        //insert ware info
        String ware_Number=TextBox1.Text;
        String ware_Name=TextBox2.Text;
        String ware_Weight=DropDownList1.SelectedItem.ToString();
        String ware_Stock=TextBox3.Text;
        String ware_Level3=DropDownList2.SelectedValue.ToString();
        String ware_Price=TextBox4.Text;
        String extend_ID=IID.ToString();
        String ware_Image=Image1.ImageUrl;
        String sql3=String.Format("insert into T_Ware
            values('{0}','{1}','{2}',{3},{4},{5},{6},'{7}')",
            ware_Number, ware_Name, ware_Weight, ware_Stock, ware_Level3,
            ware_Price, extend_ID, ware_Image);
        cmd.CommandText=sql3;
        cmd.ExecuteNonQuery();
        trans.Commit();
    }
    catch(Exception ex)
    {
        trans.Rollback();
    }
    con.Close();
}
```

7.5 总结归纳

项目 7 涉及很多内容，重点讲解了 ADO.NET 编程和 GridView 的高级操作。ADO.NET 几乎在这个项目的任务 1～5 中都有所涉及，这也说明了 ADO.NET 技术是动态网站开发所必须掌握的一项技术。另外，有很多知识点前面几个项目中已经详细介绍，所以在本项目中不再过多说明。

7.6 课后习题

编程题

本项目中前面几个任务的详细实现过程已经介绍。请读者完成商品信息的编辑(具体

界面可自行设计,也可从本书配套的源代码 AdminMP. master 母版页中继承)。

7.7　同步操练

项目经理要求开发人员能进行基本信息管理,包括分店的添加和编辑(见图 7-25 和图 7-26)、客房类型的添加和编辑(见图 7-27 和图 7-28)、客房的添加和编辑(见图 7-29 和图 7-30)。



图 7-25　分店的添加

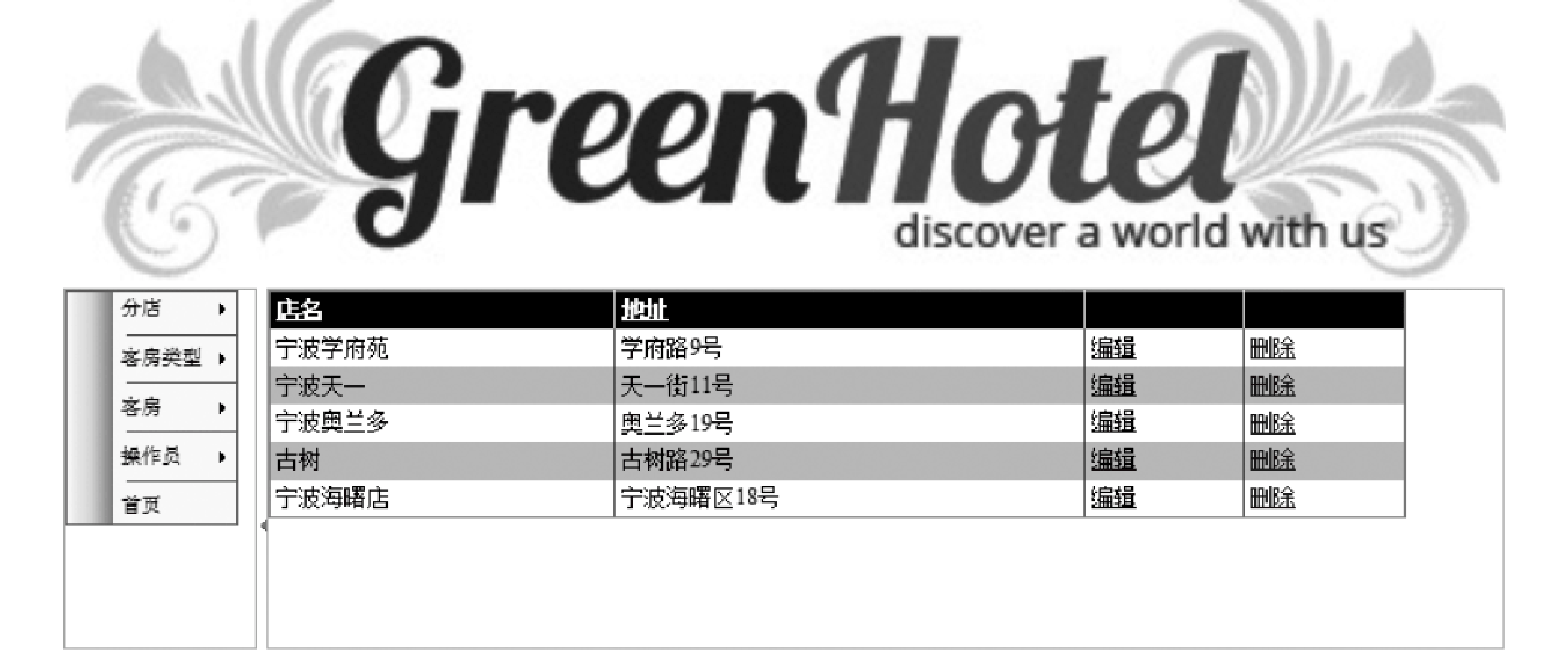
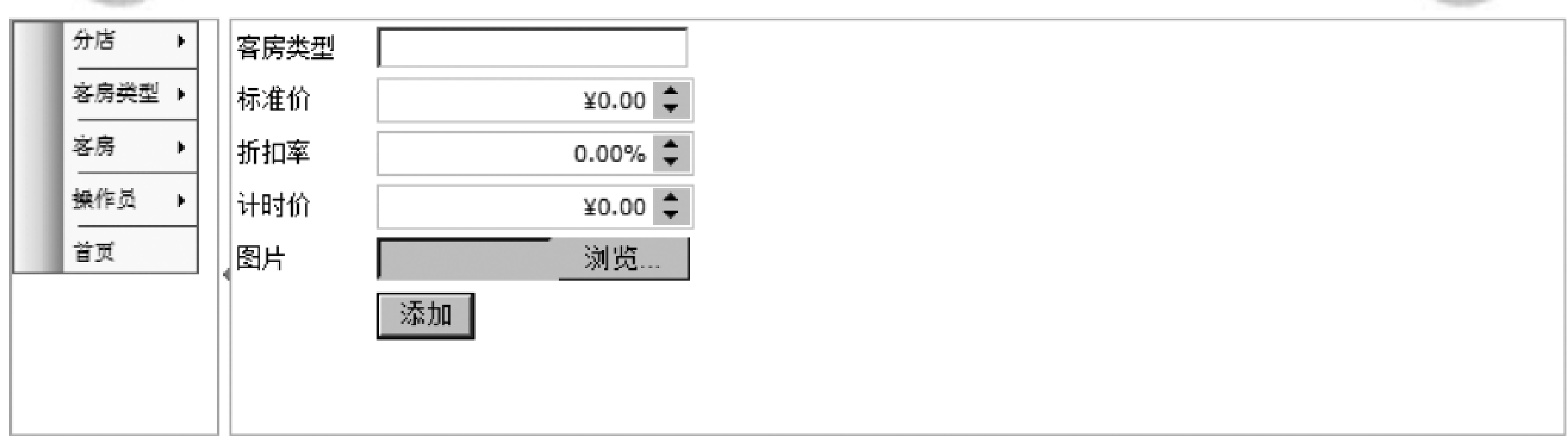


图 7-26　分店的编辑





CopyRight@格林酒店

图 7-27　客房类型的添加



客房类型	标准价	折扣(%)	计时价		
总统会客厅	¥20,000.00	26	¥2,000.00	编辑	删除
单人间	¥180.00	10	¥188.00	编辑	删除
标准间	¥288.00	15	¥250.00	编辑	删除
大床房	¥127.00	75	¥70.00	编辑	删除

分店
客房类型
客房
操作员
首页

CopyRight@格林酒店

图 7-28　客房类型的编辑



CopyRight@格林酒店

图 7-29　客房的添加



分店
客房类型
客房
操作员
首页

隶属	房间号	状态	房型		
宁波天一	1-101	Busy	单人间	编辑	删除
宁波奥兰多	1-119	Busy	单人间	编辑	删除
宁波奥兰多	1-220	Busy	单人间	编辑	删除
宁波奥兰多	2-220	Busy	总统会客厅	编辑	删除
宁波天一	6-504	Idle	标准间	编辑	删除
宁波海曙店	7-808	Busy	大床房	编辑	删除

CopyRight@格林酒店

图 7-30　客房的编辑

项目8　订 单 管 理

8.1　项 目 引 入

开发小组在项目 6 中已经完成了商品购买和订单提交功能,在项目 7 中实现了商品目录管理和商品信息管理的功能。现在开发订单管理模块。订单管理模块主要是对客户产生的订单进行编辑。当变更订单时,需要向客户注册的邮箱账户发送邮件进行提醒。

8.2　项 目 分 析

经过小组讨论和仔细分析,为了进一步将 Web 页面开发和数据库开发分离,一是计划采用 WebService 方式提供所必需的数据访问功能,这样也能为其他应用程序提供数据。二是计划采用 ASP. NET 的 Mail。

8.3　知 识 准 备

8.3.1　使用 Web Service 提供服务

Web Service 的宗旨是创建不需要用户界面就能与其他应用程序交互的 Web 应用程序。假如你正在为一家股票投资公司创建网站,并不需要把不同证券交易所的数据库与自己的后台数据库进行整合,因为你的应用程序可以使用 Web Service,并使用 XML 格式交换数据。Web Service 是松耦合的,它与服务器端和客户端使用的操作系统、编程语言都无关。创建 Web Service 必须保证的是,服务器端和客户端都要支持 HTTP、SOAP(简单对象访问协议)和 XML 等行业标准协议。

1. Web Service 是如何工作的

Web Service 允许两个程序之间交换 XML 文档。在这个架构的顶层,微软实现了一个远程过程调用(Remote Procedure Call,PRC)模型。Web Service 架构包括以下特性。

(1) WebServcie 的服务器端和客户端应用程序都能够连接到互联网。

(2) 用于通信的数据格式必须遵守相同的开放标准,并且在大多数情况下,这个标准几乎总是 SOAP。

(3) 客户端和服务器端的系统是松耦合的。即 Web Service 不关心客户端和服务器端所使用的操作系统、对象模型或者编程语言。只要 Web Service 和使用 Web Service 的应用

程序都能够发送和接收遵守适当协议标准的消息即可。

Web Service 应用流程的逻辑架构如图 8-1 所示，一个 Web Service 使用者会向 Web Service 发出一个调用请求。使用者会认为它通过 Internet 直接和 Web Service 进行交流，当然这实际上是不可能的。实际上，真正的调用由代理类完成。代理类对于 Web Service 使用者来说是一个本地类。代理会处理所有的复杂架构，包括通过 Internet 发送请求到服务器、从 Web Service 取回结果并呈现给 Web Service 使用者。由于代理类之前已在消费程序中注册，所有一切工作才可以顺利进行。注册由开发消费程序的程序员完成。

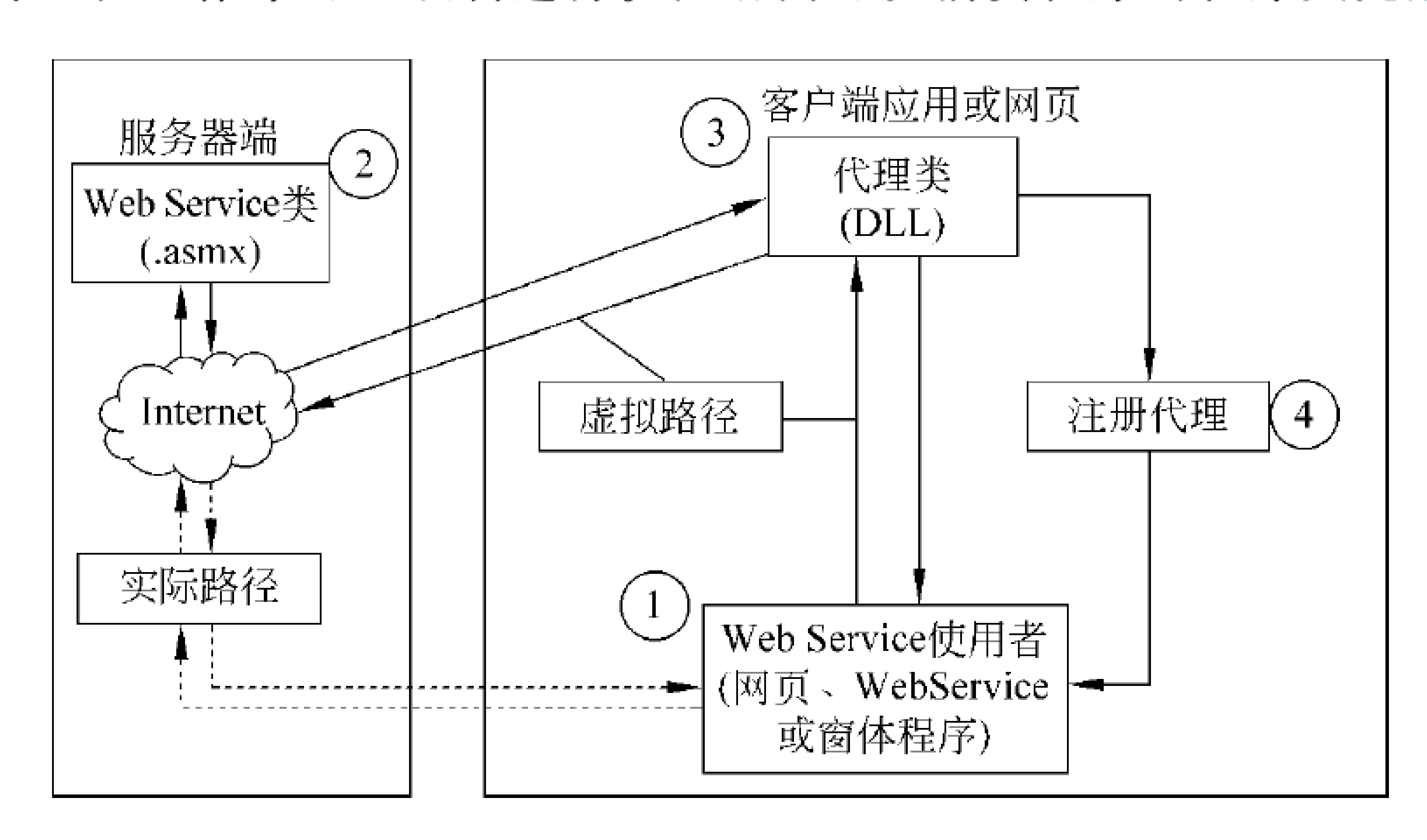


图 8-1　Web Service 应用流程的逻辑架构

客户端应用程序如果使用 Web Service，必须先创建一个代理。代理是要调用的真正代码的替身。在客户端应用程序中注册代理后，客户端应用程序调用方法时就如同调用本地对象一样。代理接受该调用，并以适当格式封装调用，然后通过 SOAP 请求发送调用程序到服务器。当服务器返回 SOAP 包给客户端后，代理会对包进行解密，并且如同调用本地对象的方法返回数据一样，代理会将这些包返回给客户端应用程序，如图 8-2 所示。

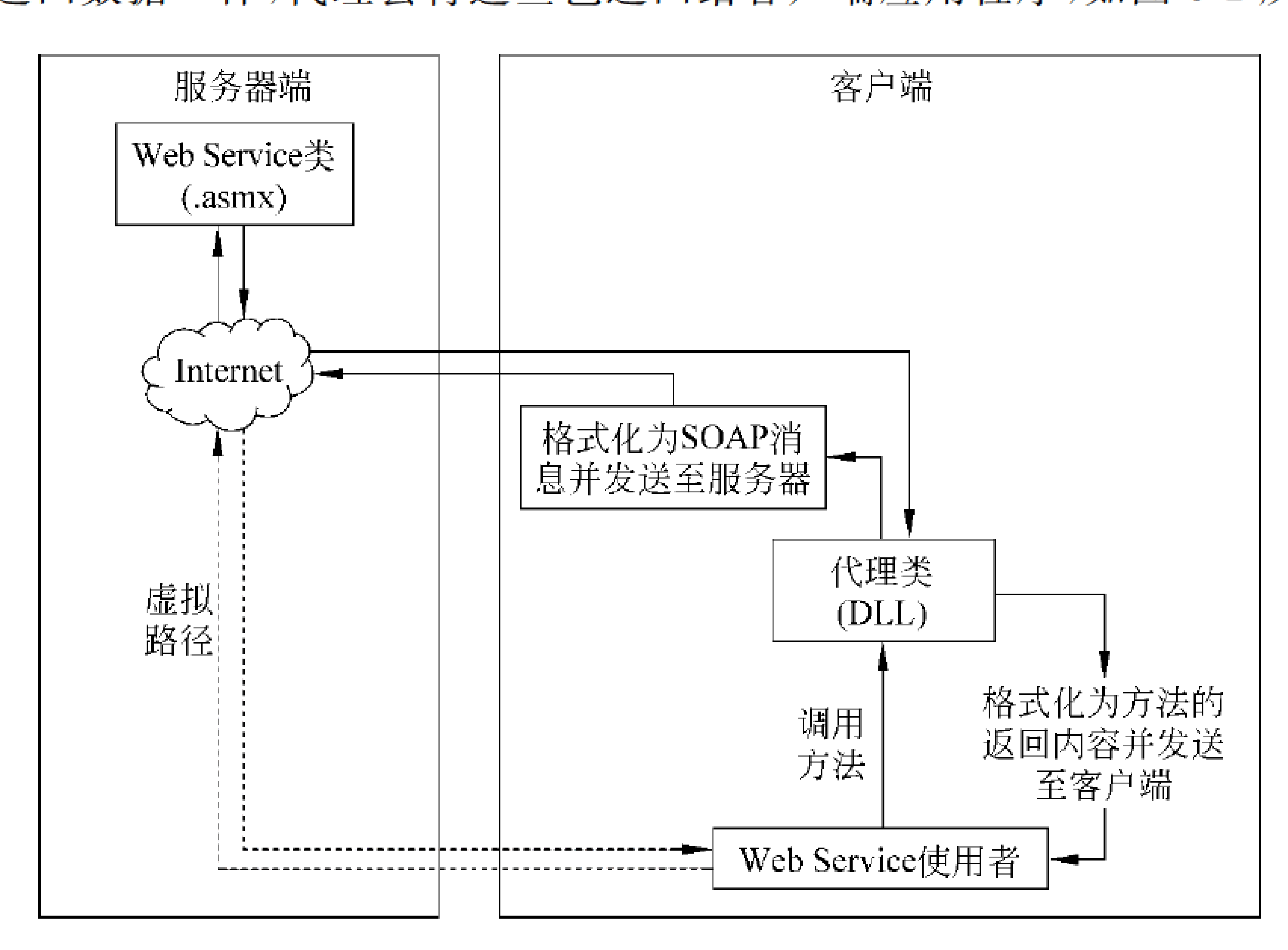


图 8-2　Web Service 代理

Web Service 使用到的协议与标准主要有 HTTP、XML、SOAP。

- HTTP：TCP/IP 协议的最上层是超文本传输协议(HTTP)。它用于在网络之间使用服务器和浏览器实现通信，主要包括在服务器和浏览器中建立连接，并将 HTML 传输到客户端浏览器。当客户端向服务器发送 HTTP 请求后，服务器就开始处理请求。通常会返回 HTML 页面，然后通过浏览器呈现。然而对 Web Service 而言，服务器返回的是 SOAP 消息，消息中包含了被调用的 Web Service 方法的返回数据。HTTP 请求从请求的浏览器处传递名称和值到服务器中，这种请求包括 GET 和 POST 两种。GET 请求中，名称和值会被附加到 URL 上，数据是未加密的。当所有需要传递的数据可以使用名称和值成对表示，而且只需要传递少量字段、字段长度也较短时，比较适合用 GET 请求。另外，如果不需考虑安全问题时，也可以使用 GET。POST 请求中，配对的名称和值是作为请求消息的一部分发送的。当有大量字段或者参数很长时，适合使用 POST 请求。从安全方面看，POST 比 GET 更安全，因为 POST 请求可以被加密。与 GET 请求一样，POST 请求无法传递复杂的数据类型(如类、结构体和 DataSet)。
- XML：是由 W3C 公布的开放标准的一种描述数据的方法。XML 和 HTML 十分类似。不同的是，HTML 使用的是预定义元素，这些元素规定了 HTML 在浏览器中如何显示，而 XML 的元素则是由开放人员自己定义的，所以几乎所有的数据都可以表示。制定 XML 的目的是使其成为一种与平台无关、语言无关的标准。XML 架构(Schema)是用于定义 XML 文档中或者 XML 文档之间元素与元素关联的文件。在架构中将指定元素名称和内容类型。HTML 与 XML 的显著差别是：大多数 HTML 读取器有很好的容错能力，而 XML 读取器则完全不同，所以 XML 文件的格式必须正确。另外，XML 元素都是小写的。
- SOAP：SOAP(Simple Object Access Protocol，简单对象访问协议)是一种用于控制数据交换的 XML 语法，它是简单的、轻量级的信息交换协议。SOAP 消息由消息内容和一个或多个头模块组成，并且封装在 SOAP Envelope 中。SOAP 使用 XML 语法来格式化内容。在设计上，SOAP 尽可能简单并且提供最小化的功能。SOAP 不需 HTTP 的 GET 和 POST 方法，它不受“名称/值”对的限制，可以使用它来发送复杂的对象，包括 DataSet、类和其他对象。SOAP 的缺点是 SOAP 消息十分冗长，因此如果存在带宽或者传输性能的问题，建议使用 POST 或 GET 方法。

2. 开发 Web Service

Web Service 没有界面，它只有方法，有一些方法支持从客户端进行远程调用的。Web Service 文件的后缀名为 asmx。Web Service 应用程序第一次运行时，如果 Web Service 是人工编译的，并且被放在虚拟根目录的 bin 目录下，那么代码隐藏对于内联编码有性能优势，因为 asmx 文件在 Web Service 运行时都会被编译成一个类。而 ASP. NET 4.0 在默认情况下没有这一优势，因为 ASP. NET 4.0 把源代码放在 App_Code 目录下，并且在第一次使用时进行编译。

示例 1：下面通过“股票跟踪”例子代码的讲解来解剖 Web Service 技术。这个 Web Service 提供了两个方法。

GetName：是一个 StcokSymbol 对象，返回一个字符串，代表了股票名称。

GetPrice：是一个 StcokSymbol 对象，返回一个数字，表示当前股票的价格。

① 首先新建一个项目(WebSite 也可以)，注意应选择.NET Framework 3.5，默认情况下.NET Framework 4 是没有 Web Service 选项的。选择 ASP.NET Web Service Application，将新的应用程序命名为 StockWebService，如图 8-3 所示。

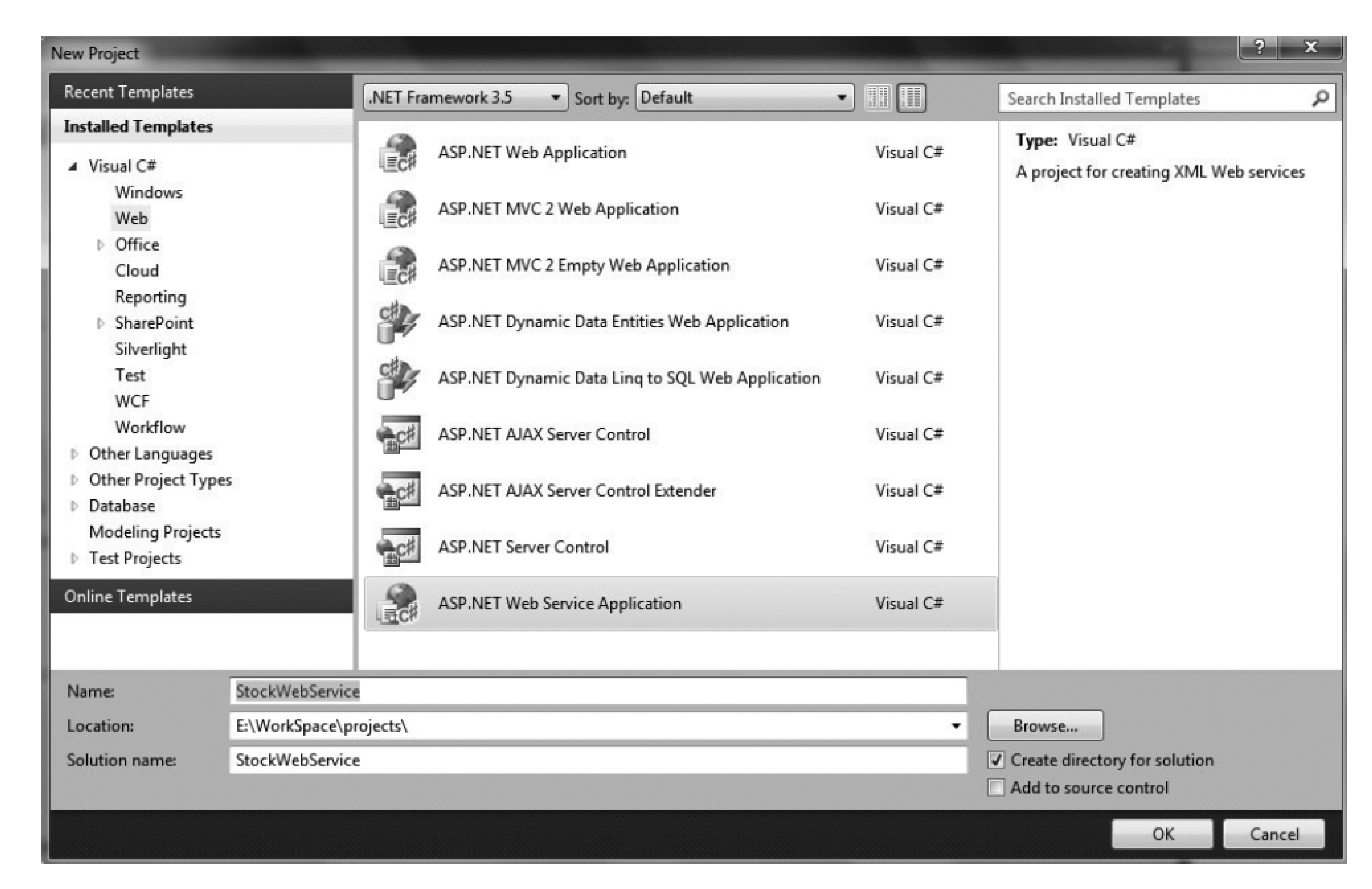


图 8-3　新建项目

可以看到，Visual Studio 2012 自动生成了如下的一些代码。

```
///<summary>
///Summary description for Service1
///</summary>
[WebService(Namespace="http://tempuri.org/")]
[WebServiceBinding(ConformsTo=WsiProfiles.BasicProfile1_1)]
[System.ComponentModel.ToolboxItem(false)]
//To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
//[System.Web.Script.Services.ScriptService]
public class Service1 : System.Web.Services.WebService
{
     [WebMethod]
     public string HelloWorld()
     {
          return "Hello World";
     }
}
```

在 Service 类中有一个名为 HelloWorld 的模板方法，它将返回一个字符串。这个方法使用 WebMethod 特性做修饰，表示该方法对 Web Service 程序为可用(WebMethod 特性会在后面解释)。按 F5 功能键运行程序，可以看到结果如图 8-4 所示。

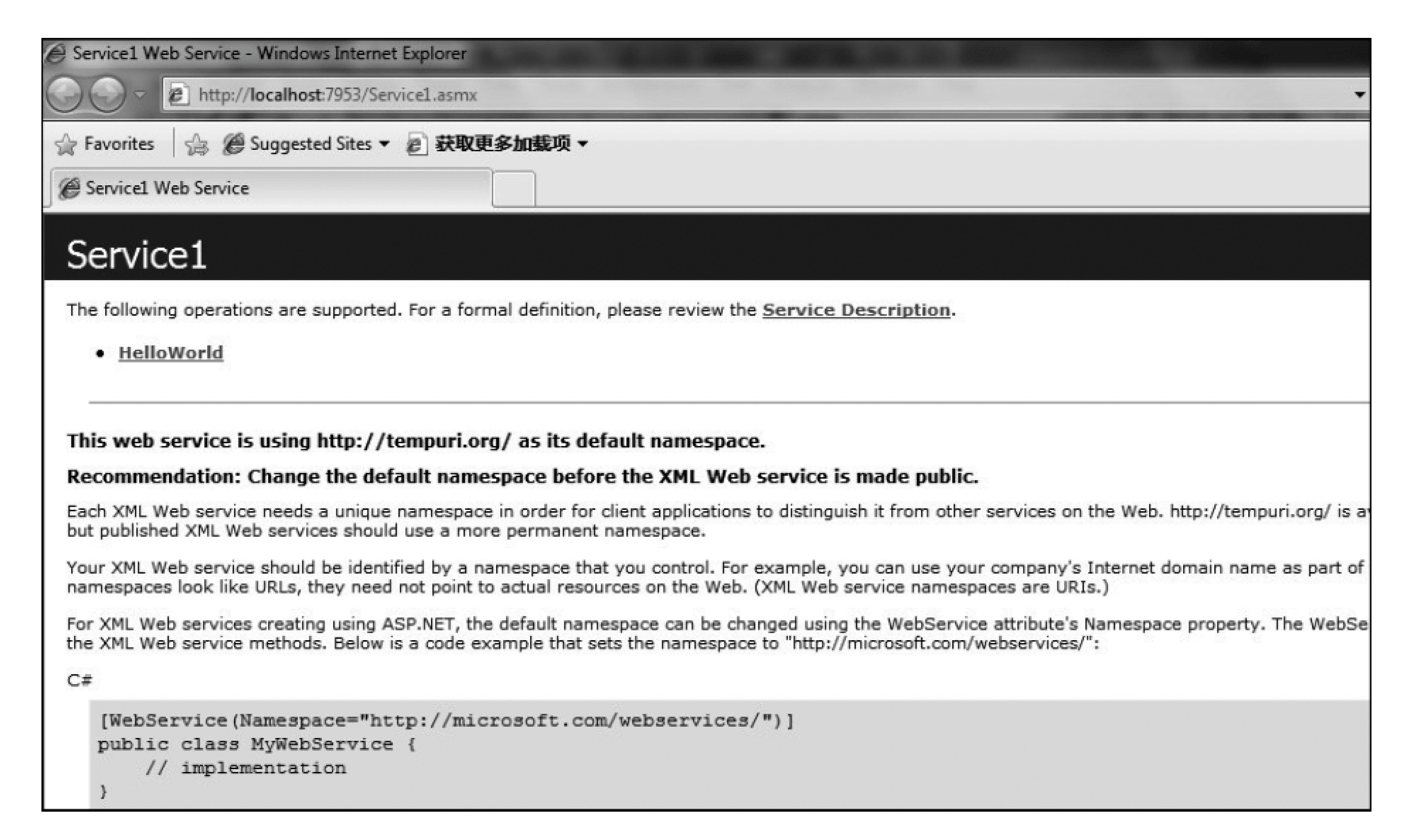


图 8-4　Hello World 的运行结果

② 代码页中输入下列代码。

```
[WebMethod]
public double GetPrice(string stockSymbol)
{
    for(int i=0; i<stocks.GetLength(0); i++)
    {
        if(string.Compare(stockSymbol, stocks[i, 0], true)==0)
            return Convert.ToDouble(stocks[i, 2]);
    }
    return 0;
}
[WebMethod]
public string GetName(string stockSymbol)
{
    for(int i=0; i<stocks.GetLength(0); i++)
    {
        if(string.Compare(stockSymbol, stocks[i, 0], true)==0)
            return stocks[i, 1];
    }
    return "Symbol not found.";
}
```

重新运行项目，发现代码中多了两个方法调用的入口。

普通的.aspx 文件把 Page 指令作为第一行代码，而 Web Service 中有一个如下的 WebService 指令（在 Visual Studio 集成开发环境下不能直接看到，可以找到 Service1.asmx 文件右击并选择“编辑”命令查看）：

```
<%@ WebService Language="C#" CodeBehind="Service1.asmx.cs"
                        Class="StockWebService.Service1" %>
```

Language 指定 Web Service 中使用的语言，不是必需的。Class 指定 Web Service 的类

名称，是必需的。WebService 指令的 CodeBehind 属性则用于指定实现 WebService 类的源代码文件的名称。Debug 如果设为 true，将以启用调试的方式编译 Web Service。默认为 false。

3. WebMethod 特性

Web Service 是由 WebService 类定义的，对于 WebService 类而言，并不需要将所有的方法提供给 Web Service 使用者。对需要提供的方法必须声明为 public，并且在方法声明之前设置 WebMethod 特性(即加上[Web Service])。WebMethod 包含一些属性，用于设定 Web 方法的行为。语法为[WebMethod(PropertyName=value)]，以下是一些属性的说明。

- BufferResponse：默认情况下，ASP.NET 在从服务器端向客户端发送响应之前，会对整个响应进行缓存。大多数情况下，这是最好的做法。但是，如果响应时间非常长，那么需将它设为 true 来禁用缓存。如果设为 false，则返回到客户端的响应是 16KB 的块。默认值为 true。
- CacheDuration：如同网页，Web Service 也能够把返回到客户端的结果进行缓存(如果客户端发出的请求与由另一个客户端发出的请求完全一致是，那么服务器就返回缓存中的响应。可以改善程序的性能)。CacheDuration 定义第一次请求之后多少秒内，会在响应中将缓存页被发送给随后的请求。一旦缓存过期，则发送新页面给请求。默认值为 0，即禁用结果缓存。如果 Web 方法返回的数据没有改变(如，从一小时更新一次的数据库中获取数据的查询)，那么就可以设置结果并缓存为一个合适的时间，例如 1800s。另外，如果返回的数据是动态的，那么需要设置缓存持续时间较短或干脆禁用。如果 Web Service 没有一个相对有限的参数范围，也不适合使用缓冲。
- Description：对 Web 方法的描述，是字符串类型。
- EnableSession：默认为 false。如果设为 true，Web 方法将启用会话状态。如果设为 ture 且 Web Service 继承自 WebService 类，那么会话可以使用 WebService.Session。允许会话状态为应用程序增加额外的开销。
- MessageName：在 C# 类中，方法可以拥有相同的名字(重载)，Web Service 禁止使用重载。WebMethod 特性的 MessageName 属性可以消除由多个相同名称造成的无法识别的问题。它允许对每一个方法的重载使用唯一的别名。当重载方法在 SOAP 消息中引用时，SOAP 消息将使用 MessageName 而非方法的名称。
- TransactionOption：ASP.NET Web 方法可以使用事务，但是仅当事件在 Web 方法中初始化时可以使用。TransactionOption 属性用于设置 Web 方法是否启动一个事务。然而，因为 Web 方法的事务必须为根对象，所有只有两个不同的行为，即启用一个新对象(Required、RequiresNew)或者不启动(Disabled、NotSupported、Supported)。

4. WebService 特性

不要把它与 WebMethod 混淆，WebService 特性允许向 Web Service 添加额外的信息。语法为[WebService(PropertyName=value)]。如果有多个属性，可以使用逗号分隔。下面是它的一些属性。

- Description：描述 Web Service。

- Name：当在浏览器中测试页面时，在 Web Service 帮助页面的最顶部会显示 Web Service 的名称，且对所有潜在的 Web Service 使用者可用。默认情况下，Web Service 的名称是实现 Web Service 类的名称。
- Namespace：每个 Web Service 都有一个 XML 命名空间。XML 命名空间允许在 XML 文档中创建名称，这个名称是一个统一资源标识符（URI）。Web Service 使用在 XML 中定义的 WSDL 文档进行描述。每个 Web Service 特性必须有一个独立的 XML 命名空间，这样它才能够为应用程序唯一识别。Visual Studio 创建的 Web Service 的默认 URI 为 http://tempuri.org。通常使用唯一的名称来定义一个新的命名空间，例如公司的网站（可以不是一个有效的 URL）。

5. 数据类型

Web Service 可以使用任何 CLR 支持的数据类型作为参数或者返回值。除了基本数据类型以外，还可以使用数组和基本类型的 ArrayList。由于数据是通过 XML 在 Web Service 和客户端之间传递的，那么无论使用参数或返回值，都必须使用 XML Schema 或者 XSD 表示。

此外，Web Service 能够把用户定义的类和结构体作为参数或者返回值，这里有一些需要记住的规则，内容如下。

- 所有类变量必须是基本数据类型或者基本数据类型的数组。
- 所有类变量必须是公开的或者有一个公开的实现 GET 和 SET 访问器的属性。
- DataSets：Web Service 能够通过 XML 编码后返回任何数据，这也包括返回 DataSet，这是因为 ADO.NET 内部使用 XML 来表示 DataSet。一个 DataSet 仅是 ADO.NET 数据存储中的一种可以由 Web Service 返回的类型而已。

6. 创建发现文档

一旦完成 Web Service 的创建，负责开发 Web Service 程序的程序员需要找到一种方法能够了解服务器上有哪些 Web Service 可用，以及这些 Web Service 提供了哪些方法，这些方法和属性可接收哪些参数，这些 Web 方法返回的值是什么。这个过程叫“发现”，是可选的。如果 Web Service 使用程序的开发者了解这些 Web Service 文件的 URL，那么就不需要实施“发现”动作了。可以使用 disco.exe 在命令行方式下创建 XML 文件（即发现文档）。

打开 Visual Studio 命名提示符，输入如下代码：

```
disco /out:<输出目录名称>http://localhost:7953/Service1.asmx
```

可以通过以下两种方法寻找“发现”文档：

- 通过查询字符串。即在 service.asmx 文件后加上？disco，如 http://localhost:7953/Service1.asmx?disco。
- 静态发现文件。如果 Web Service 的程序需要使用静态发现文件，那么 Web Service 开发人员必须创建一个静态发现文件。

“发现”是一个过程，它是一个可选过程，可以在客户端机器上从命令行执行 disco 工具，并把 Web Service 的 URL 作为参数传递给它。如 disco http://localhost:7953/Service1.asmx，这个命令通过指定 URL 来寻找一个发现文档，并把它们保存在本地计算机的当前目录中。还有一个.wsdl 文件也将在当前目录中生成并保存。也可以使用“/out:”

参数将输出目录改为指定的目录，如，disco /out：C：/Temp/http：// localhost：7953/Service1. asmx，执行该命令会把两个文件生成到输出目录的 Service1. wsdl 文件中，这个文件与通过在浏览器中输入. asmx?wsdl 或者使用 wsdl 命令生成的 wsdl 是相同的。

7. 部署

部署 Web Service 与部署网页差不多。. asmx 文件必须位于 IIS 提供的虚拟目录中，这样它才可以被浏览器访问到。如果有一个 Web Service 的. disco 文件，那么这个文件也必须放在应用程序虚拟目录下。同样，如果应用程序需要创建一个 web. config 文件，那么也必须将其复制到应用程序虚拟目录中。与使用网页一样，对于已编译的类和资源，既可以使用预编译的 assemblies，也可以使用动态编译的 assemblies 来处理。

8. 使用 VS 创建客户端

示例 2：下面用一个示例来创建客户端程序并调用前面章节中创建的 Web Service（即 StockWebService）。

① 新建一个 ASP. NET 应用程序，右击解决方案，单击 Add Web Reference 菜单项，如图 8-5 所示，打开 Add Web Reference 对话框，如图 8-6 所示。

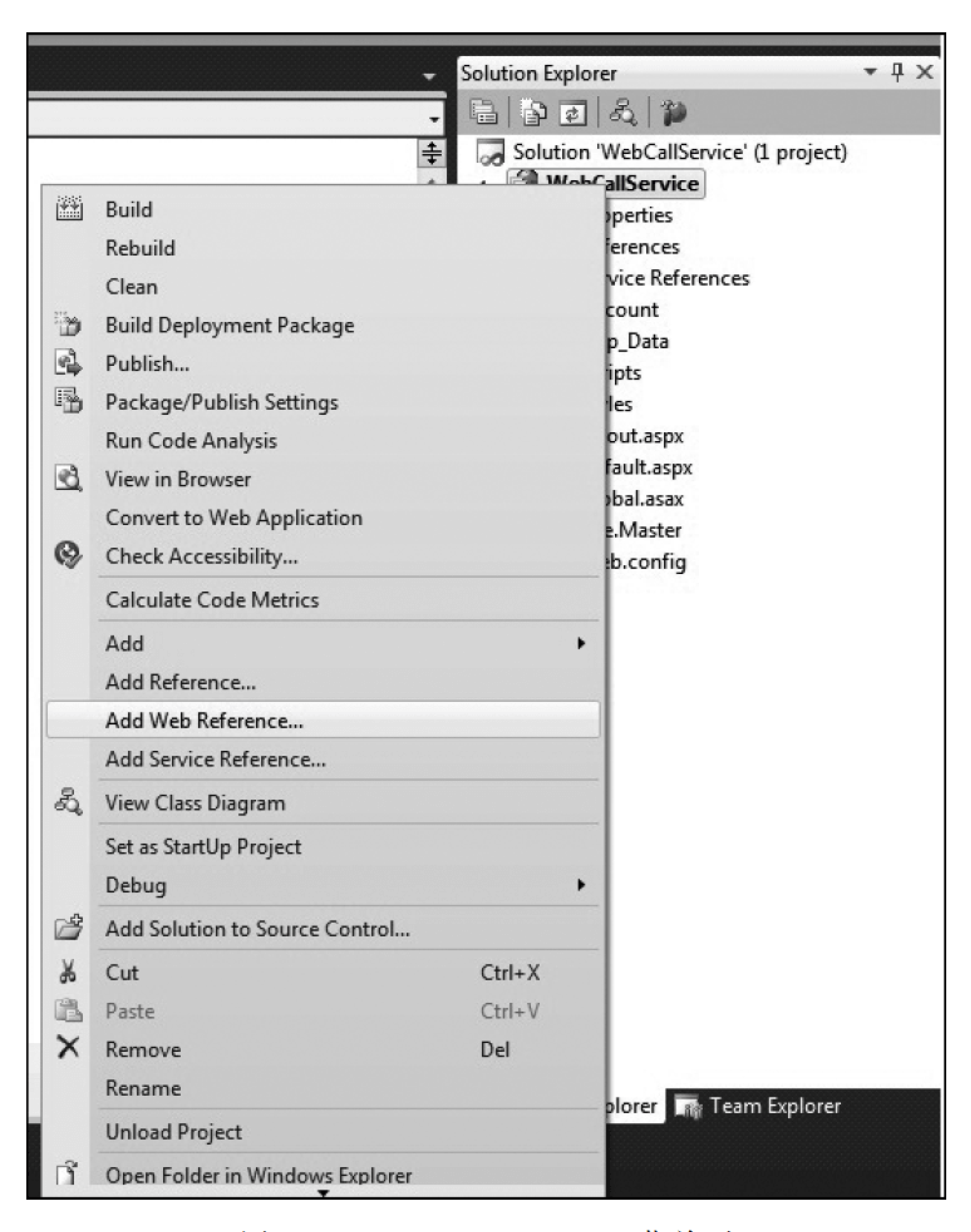


图 8-5　Add Web Reference 菜单项

在 URL 文本框中输入 http://localhost:7953/Service1. asmx，即 Web Service 的地址，注意此时 Web Service 需处于运行状态，单击右边的箭头，会将该 Web Service 中的方法查

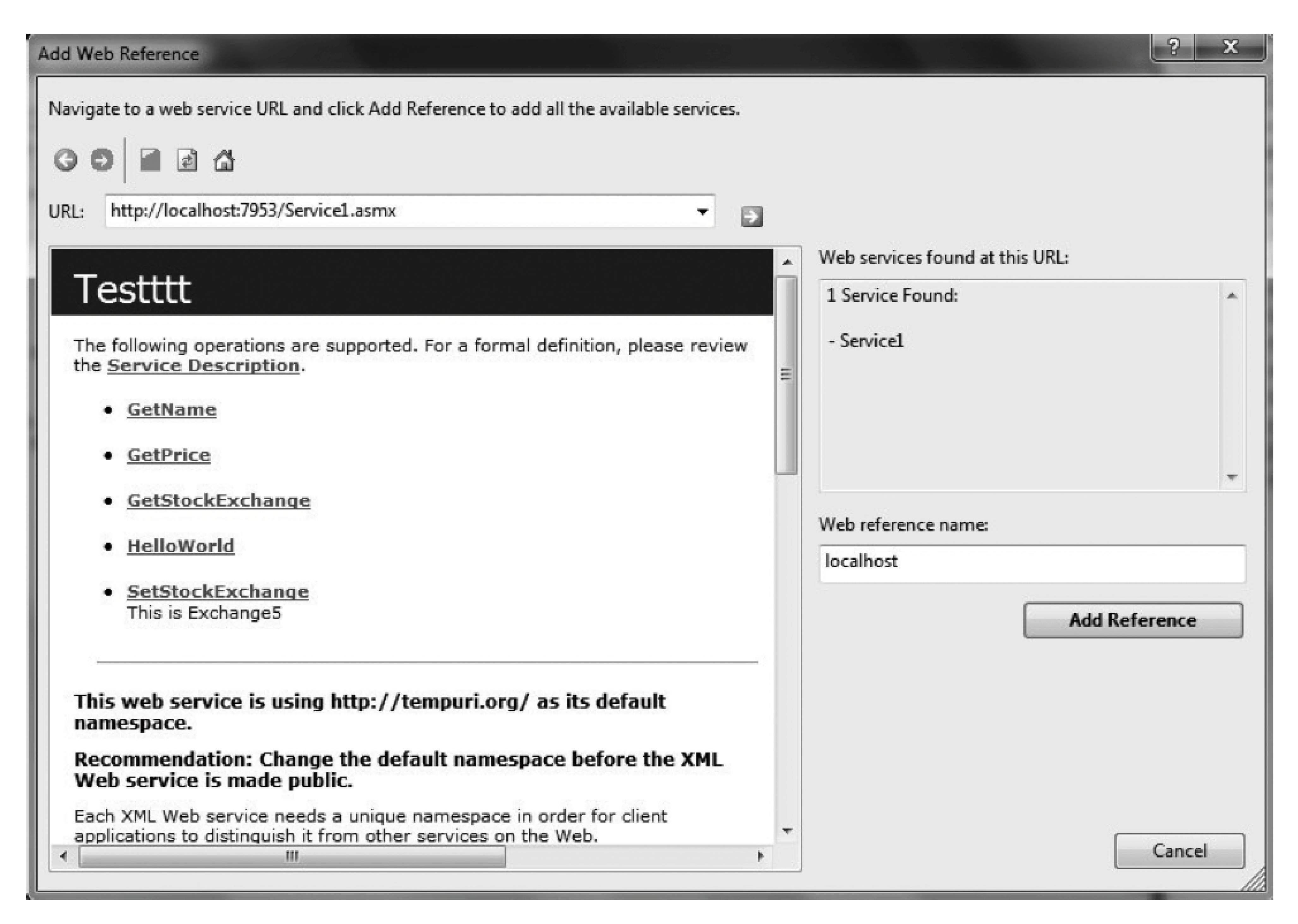


图 8-6　Add Web Reference 对话框

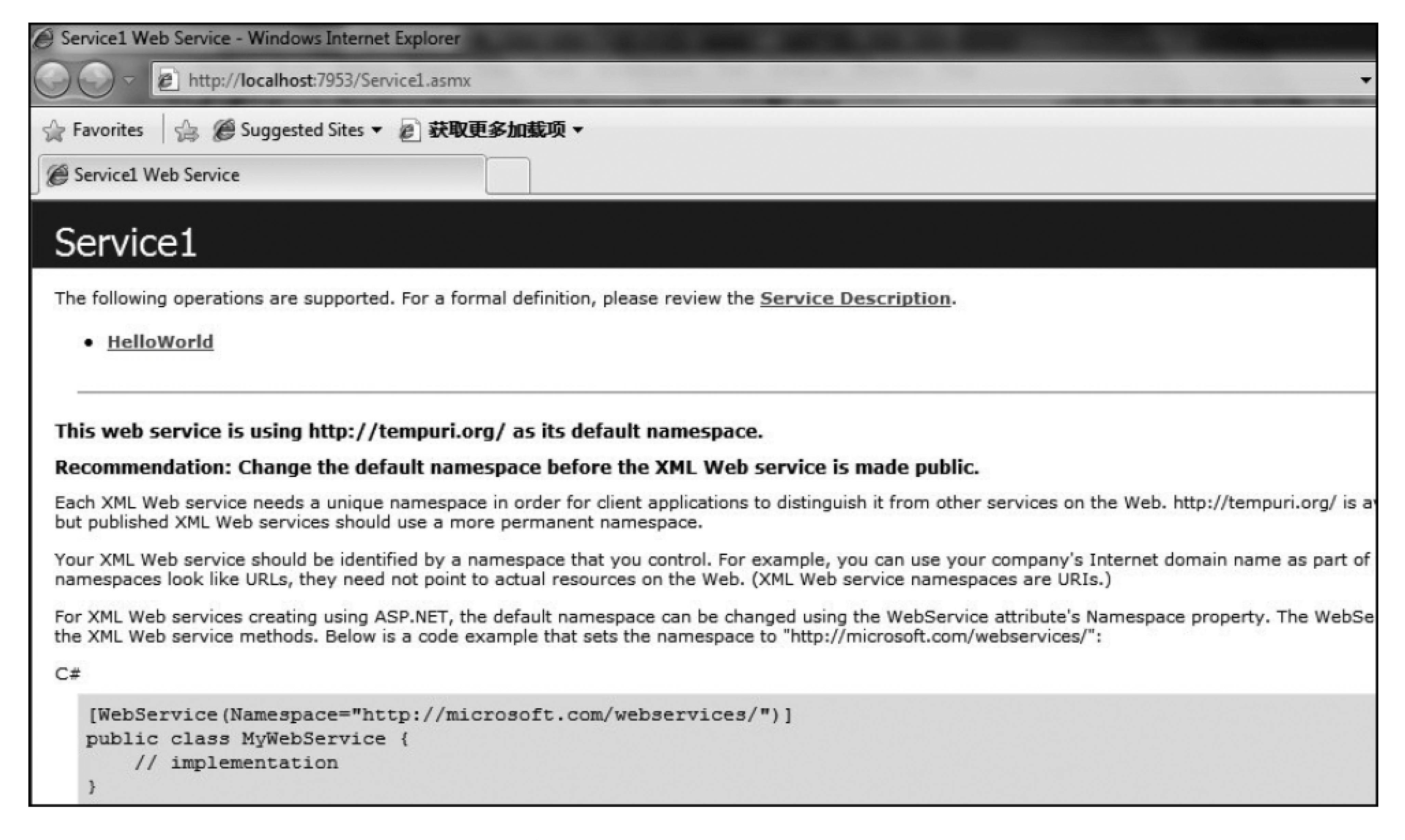


图 8-7　调用 Web Service

询出来，我们可以单击某个方法来了解具体的信息，如单击 GetName，则会出现调用的参数及 SOAP、POST、GET 对应的 XML 文档，如图 8-7 所示。

填写好 Web 引用的名字后，就可以添加 Web 引用了。此时会发现 ASP. NET 应用程序中多了一些文件，此时的目录结构如图 8-8 所示。

② 双击刚刚添加的 Web 引用的名称 localhost，打开对象浏览器，可以看到 Web Service 项目下的一些事件及参数，其中，“方法名＋Completed＋EventArgs”表示事件参数，



图 8-8　Web Service 相关的目录结构

"方法名＋Completed＋EventHandler"表示事件处理方法。

查看 ASP. NET 应用程序的目录，发现生成了两个目录 Service References 和 Web References，其中 Web References 目录下存放的是刚才添加的 Web 引用，里面的四个文件为：Reference. cs、Reference. map、Service1. disco 和 Service1. wsdl。并且在 web. config 中自动添加了如下的代码。

```
<applicationSettings>
    <WebCallService.Properties.Settings>
        <setting name="WebCallService_localhost_Testttt" serializeAs="String">
            <value>http://localhost:7953/Service1.asmx</value>
        </setting>
    </WebCallService.Properties.Settings>
</applicationSettings>
```

然后在后台代码中就可以调用 Web Service 了。注意有一个类为 Testttt，这是我们重命名的一个 Service。后台调用的代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    localhost.Testttt proxy=new localhost.Testttt();
    Response.Write("web:"+proxy.GetName("a"));
}
```

即在页面中放一个按钮，单击该按钮后返回 Web Service 的名称，这样运行的结果为"web:Symbol not found."

8.3.2　使用 SMTP 发送电子邮件

说到发送邮件，先提一下 SMTP。SMTP 的全称是 Simple Mail Transfer Protocol，即简单邮件传输协议。它是一组用于从源地址到目的地址传输邮件的规范，通过它可以来控制邮件的中转方式。SMTP 协议属于 TCP/IP 协议簇，它帮助每台计算机在发送或中转信件时找到下一个目的地。SMTP 服务器就是遵循 SMTP 协议的发送邮件服务器。接下来介绍一下名称空间(NameSpace)的 System. Web. Mail 类库里所提供的邮件发送的对象、属

性和方法，它有三个类：SmtpMail、MailMessage 和 MailAttachment。MailMessage 类提供属性和方法来创建一个邮件消息对象。MailAttachments 类提供属性和方法来创建一个邮件附件对象。SmtpMail 类提供属性和方法并通过使用 windows 2000 CDOSYS 的消息组件的联合数据对象来发送邮件消息。MailMessage 对象的常见属性见表 8-1。

表 8-1　MailMessage 对象的常见属性

From	发送邮件的地址
To	接受邮件的地址
Subject	邮件的标题
Priority	邮件的优先级(有效值为 High、Low、Normal)
Attachments	返回一个集合,代表附件
Bcc	密送地址
Cc	抄送地址
Body	获取或是设置电子邮件消息的内容
BodyFormat	获取或是设置 MailFormat 的枚举值,此值指定消息体邮件的格式(HTML 格式、Text 格式)
Bodyencoding	指定消息的编码方式(主要有 Base64、UUencode)

这里解释一下密送和抄送的区别：密送就是你群发邮件后,收邮件的人无法看到你发给了多少人以及他们的邮件地址;抄送就是群发邮件时,收邮件的人可以看到你发给了多少人以及他们的邮件地址。SmtpMail 类的 Send 方法的作用就是发送邮件,有两个重载方法。

- SmtpMail. Send("发送邮件的地址","接收邮件的地址","邮件的标题","邮件消息的内容")：这个方法很简单,不适合发送带附件的邮件。
- SmtpMail. Send(MailMessage)：此方法复杂、灵活,适合发送附件,而且可以设置 MailMessage 对象的各种属性值。

如果用 ASP. NET 开发一个邮件发送的程序,那么首先应该得到 SMTP。有两种方法：第一种方法调用目前知名的邮件服务提供商的 SMTP,比如新浪、搜狐、网易的免费电子邮箱的 SMTP;第二种方法是本地装一个 SMTP 虚拟服务器。本书只介绍用目前知名的邮件服务提供商的 SMTP 来发送电子邮件的方法。

在 ASP. NET 中利用知名的邮件服务提供商的 SMTP 来发送邮件,必须要有相应邮件服务提供商的注册账号。如果没有,首先需要去相应的邮件站点上注册免费邮箱,因为要使用邮件服务提供商的 SMTP,他们需要对注册人员的身份进行验证,这样可以避免产生大量的垃圾邮件。假设在新浪的邮件站点(mail. sina. com. cn)上注册了一个免费电子邮件,用户名是 mysina,密码是 chenjie。该账号为虚构的,应使用自己注册的用户名称和密码代替。新浪的邮件站点的 SMTP 地址是：smtp. sina. com. cn。现在需要向 scucj@126. com 发送邮件。那么利用 ASP. NET(C＃)发送邮件的核心代码如下。

```
using System.Web.Mail;
MailMessage objMailMessage;
MailAttachment objMailAttachment;
//创建一个附件对象
objMailAttachment=new MailAttachment("d:\\test.txt");
```

```
//创建邮件消息
objMailMessage= new MailMessage();
objMailMessage.From= "mysina@ sina.com";
objMailMessage.To= "scucj@ 126.com";
objMailMessage.Subject= "邮件发送标题:你好";//发送邮件的标题
objMailMessage.Body= "邮件发送标内容:测试一下是否发送成功!";
objMailMessage.Attachments.Add( objMailAttachment);
objMailMessage.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
    smtpauthenticate","1");
objMailMessage.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
    sendusername", "mysina");
objMailMessage.Fields.Add("http://schemas.microsoft.com/cdo/configuration/
    sendpassword", "chenjie");
SmtpMail.SmtpServer= "smtp.sina.com.cn";
SmtpMail.Send( objMailMessage);
```

如果没有上述粗体标注的三行代码,则会出现如下错误提示:服务器拒绝了一个或多个收件人的地址。服务器响应为:554 Client host rejected:Access denied。

8.4 项目实施

8.4.1 任务 1:以 Web Service 方式提供访问 Smart 数据库的服务

1. 任务目标

能创建 Web Service 方法。

2. 任务内容

创建提供 Smart 数据库访问的 Web Service。

3. 任务实施步骤

(1) 创建 Web Service。打开 Visual Studio 2012 开发工具,单击"文件"→"新建"→"网站"菜单项,打开"新建网站"对话框。单击.NET Framework 版本下拉列表,选择.NET Framework 3.5。单击"ASP.NET Web 服务"列表项,在"Web 位置"文本框中输入 SmartADOService,单击"确定"按钮完成操作。

(2) 右击 Service.asmx 文件,单击"查看代码"菜单项,打开 Service.cs 代码页,编写访问数据库的服务方法,当然不可少的是需要使用 using System.Data.SqlClient;using System.Data 导入命名空间。编写 Web Service 方法时需要注意两点:一是比普通的方法要多一个[WebMethod]特征;二是返回的数据类型是可序列化的,例如 DataSet。代码如下。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using System.Data.SqlClient;
```

```
using System.Data;
[WebService(Namespace="http://tempuri.org/")]
[WebServiceBinding(ConformsTo=WsiProfiles.BasicProfile1_1)]
//若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务,请取消以下行的注释
//[System.Web.Script.Services.ScriptService]

public class Service : System.Web.Services.WebService
{
    String connectionString="Data Source=.;Initial Catalog=Smart; Uid=sa;
    Pwd=zzb;
        MultipleActiveResultSets=true";
    public Service()
    {
        //如果使用设计的组件,请取消以下行的注释
        //InitializeComponent();
    }

    [WebMethod]
    public DataSet  QueryOperation(String StrQueryCommand)
    {
        SqlConnection con=new SqlConnection(connectionString);
        if(con.State==System.Data.ConnectionState.Closed)
            con.Open();
        SqlCommand cmd=new SqlCommand();
        cmd.Connection=con;
        cmd.CommandText=StrQueryCommand;
        DataSet ds=new DataSet();
        SqlDataAdapter sda=new SqlDataAdapter(cmd);
        sda.Fill(ds);
        return ds;
    }
    [WebMethod]
    public bool ExeNonQuery(String StrCmd)
    {
        SqlConnection con=new SqlConnection(connectionString);
        if(con.State==System.Data.ConnectionState.Closed)
            con.Open();
        SqlCommand cmd=new SqlCommand();
        cmd.Connection=con;
        cmd.CommandText=StrCmd;
        try
        {
            cmd.ExecuteNonQuery();

        }
        catch(Exception ex)
        {
            throw ex;
            return false;
        }
        return true;
```

```
        }
        [WebMethod]
        public bool ExeNoQueryProc(String cmdName, SqlParameter[] ps)
        {
            SqlConnection con= new SqlConnection(connectionString);
            if(con.State==System.Data.ConnectionState.Closed)
                con.Open();
            SqlCommand cmd= new SqlCommand();
            cmd.Connection= con;
            cmd.CommandText= cmdName;
            cmd.CommandType= System.Data.CommandType.StoredProcedure;
            foreach(SqlParameter p in ps)
            {
                cmd.Parameters.Add(p);
            }
            try
            {
                cmd.ExecuteNonQuery();

            }
            catch(Exception ex)
            {

                return false;
            }
            return true;
        }
        [WebMethod]
        public DataSet  QueryOperationProc(String cmdName, SqlParameter[] ps)
        {
            SqlConnection con= new SqlConnection(connectionString);
            if(con.State==System.Data.ConnectionState.Closed)
                con.Open();
            SqlCommand cmd= new SqlCommand();
            cmd.Connection= con;
            cmd.CommandText= cmdName;
            cmd.CommandType= System.Data.CommandType.StoredProcedure;
            if(ps != null)
            {
                foreach(SqlParameter p in ps)
                {
                    cmd.Parameters.Add(p);
                }
            }
            DataSet ds= new DataSet();
            SqlDataAdapter sda= new SqlDataAdapter(cmd);
            sda.Fill(ds);
            return ds;
        }
    }
```

(3) 按 F5 键运行程序，如图 8-9 所示。单击 QueryOperation 超链接(见图 8-10)，在“值”文本框中输入 select * from t_customer，单击“调用”按钮，返回最终的运行结果(见图 8-11)。

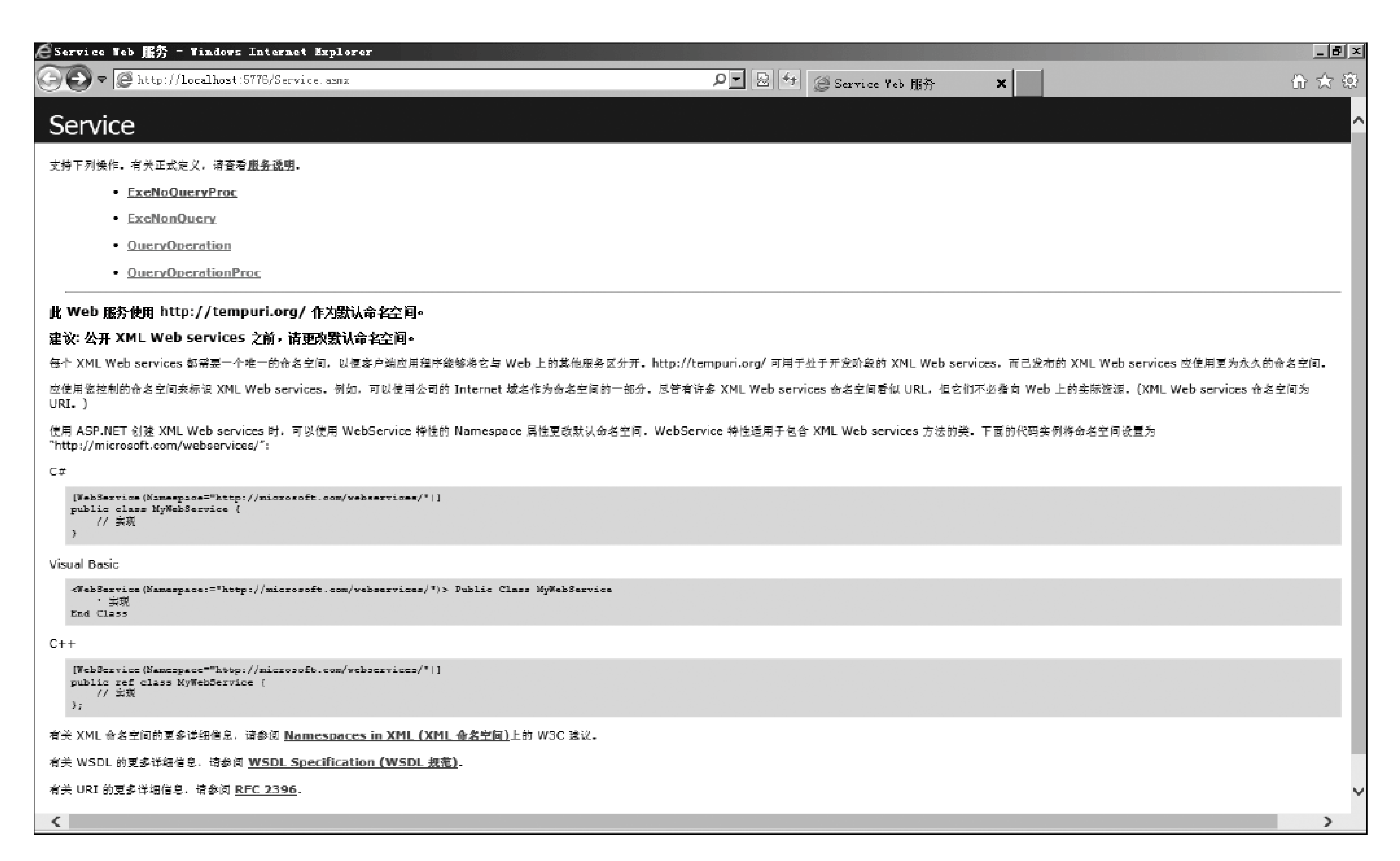


图 8-9 任务 1 运行效果

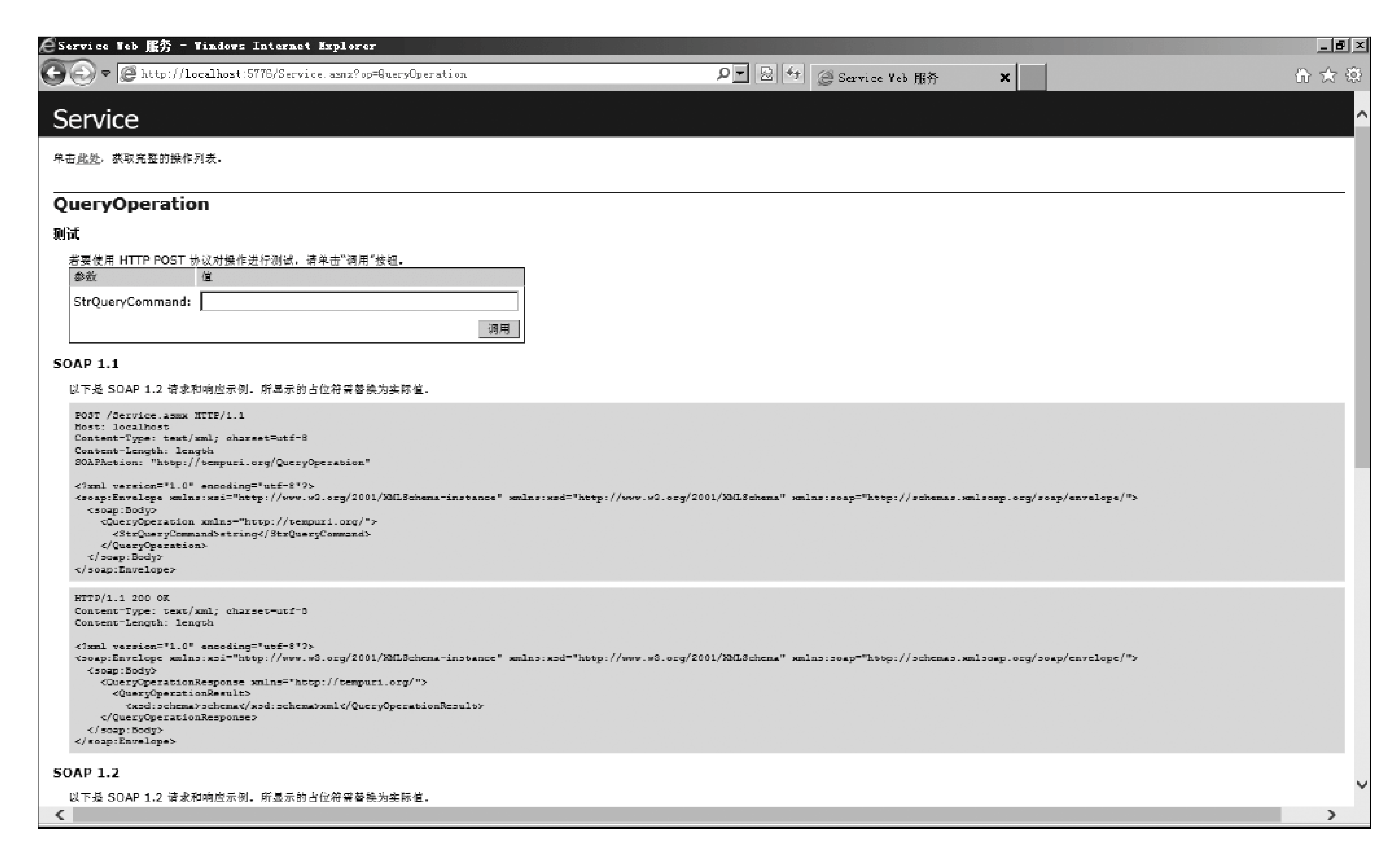


图 8-10 单击 QueryOperation 超链接

8.4.2 任务 2：实现订单的编辑

1. 任务目标

(1) 能熟练操作 GridView 控件。

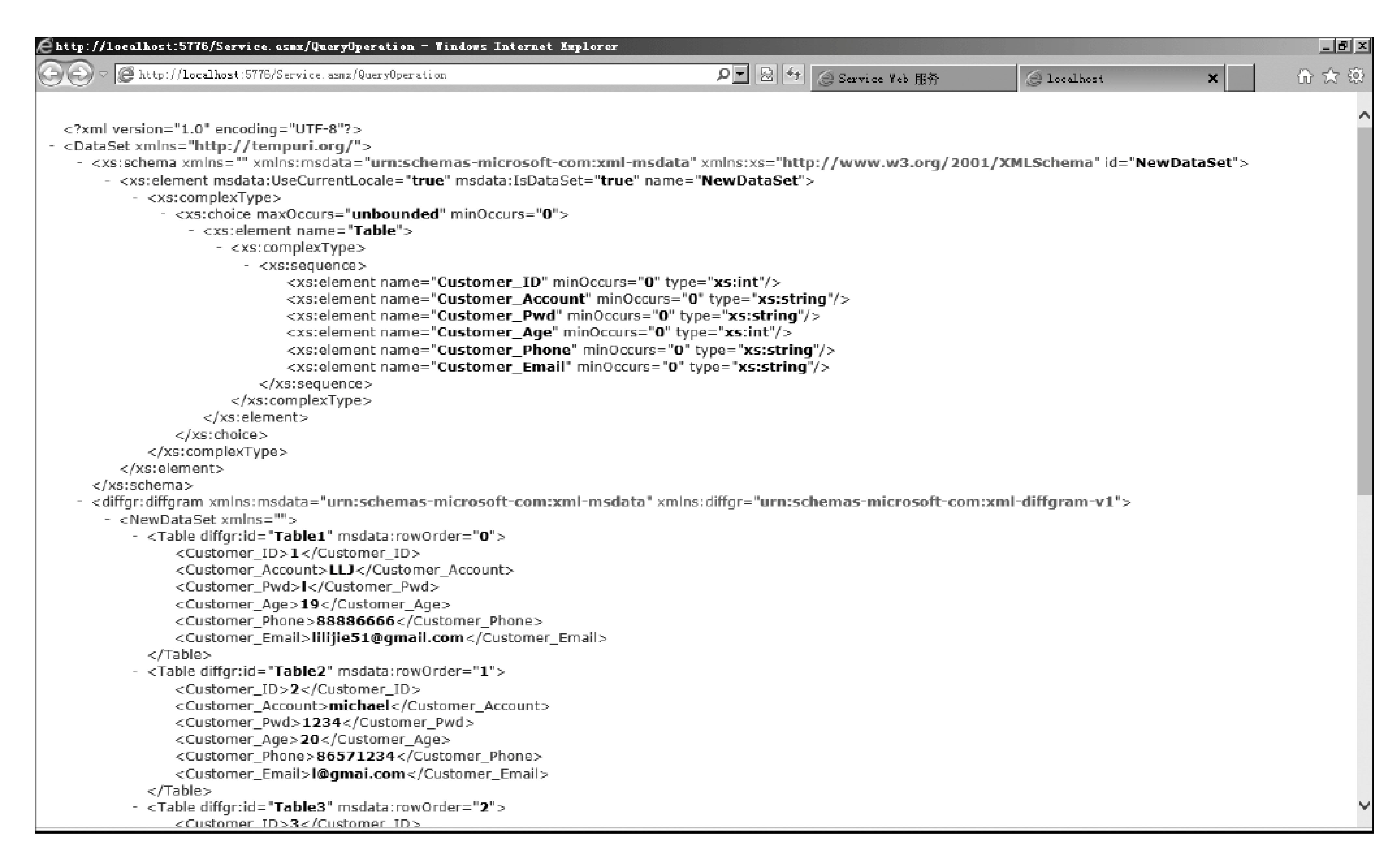


图 8-11　程序运行后返回结果

（2）能调用 Web Service。

2. 任务内容

（1）实现订单数据列表显示的功能。

（2）实现修改订单的功能

（3）实现 E-mail 通知用户的功能。

3. 任务实施步骤

（1）添加任务 1 中创建的 Web Service。打开 Smart On Line 电子商城网站，右击网站根目录，单击“添加服务引用”菜单项，打开“添加服务引用”对话框，在“地址”文本框中输入 http://localhost:5776/Service.asmx（任务 1 中创建的 Web Service 地址）。单击“转到”按钮显示所发现的服务，如图 8-12 所示。单击“确定”按钮，完成添加 Web Service 引用的操作。

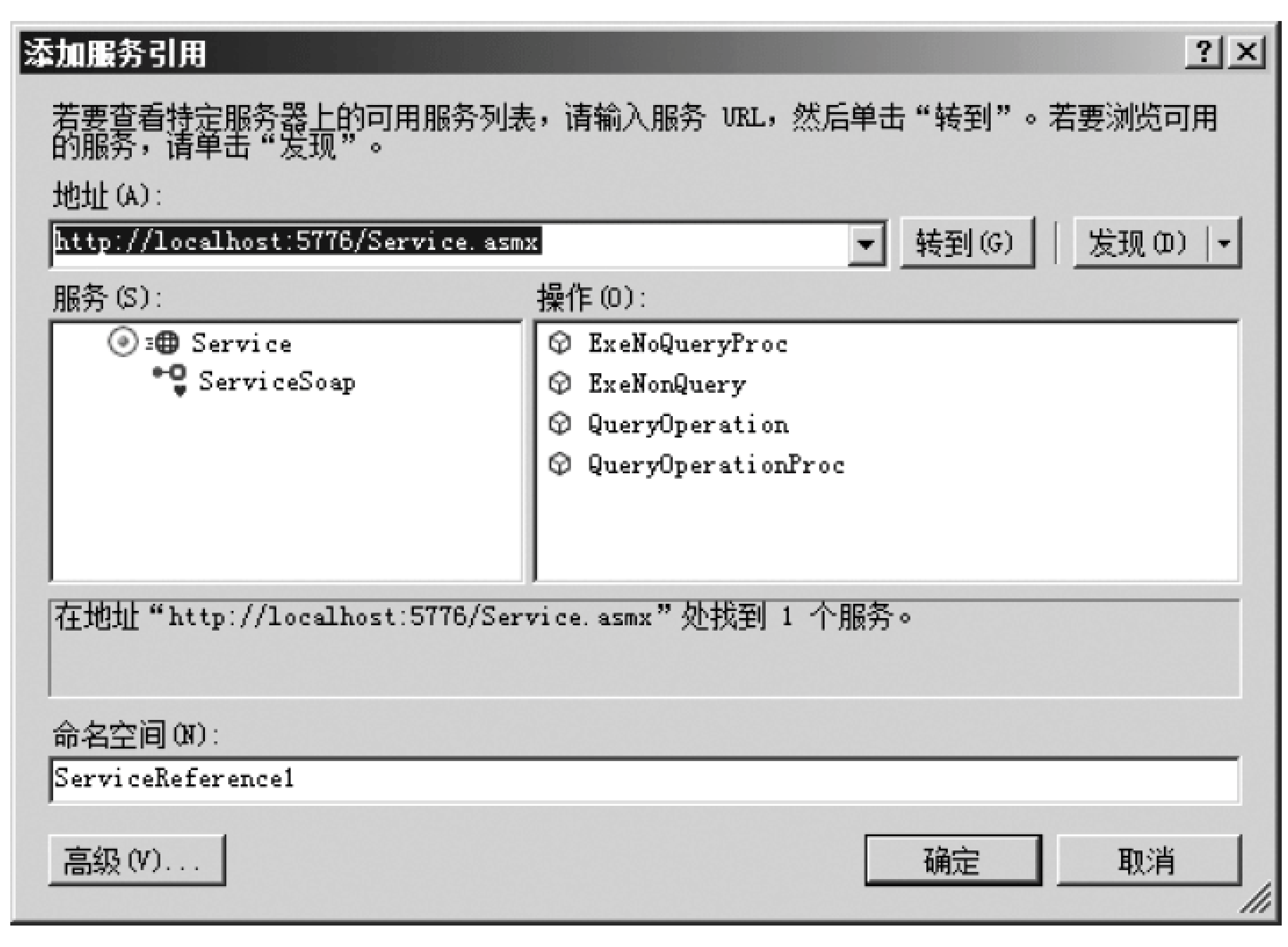


图 8-12　“添加服务引用”对话框

(2) 新建 EditOrder 内容页，选择 AdminMP. master 母版页。右击 Admin 文件夹，单击“添加”→“添加新项”命令，单击“Web 页”选项，单击“选择母版页”，“名称”文本框中输入 EditOrder. aspx，单击“确定”按钮。再单击 AdminMP. master 母版页，单击“确定”按钮。

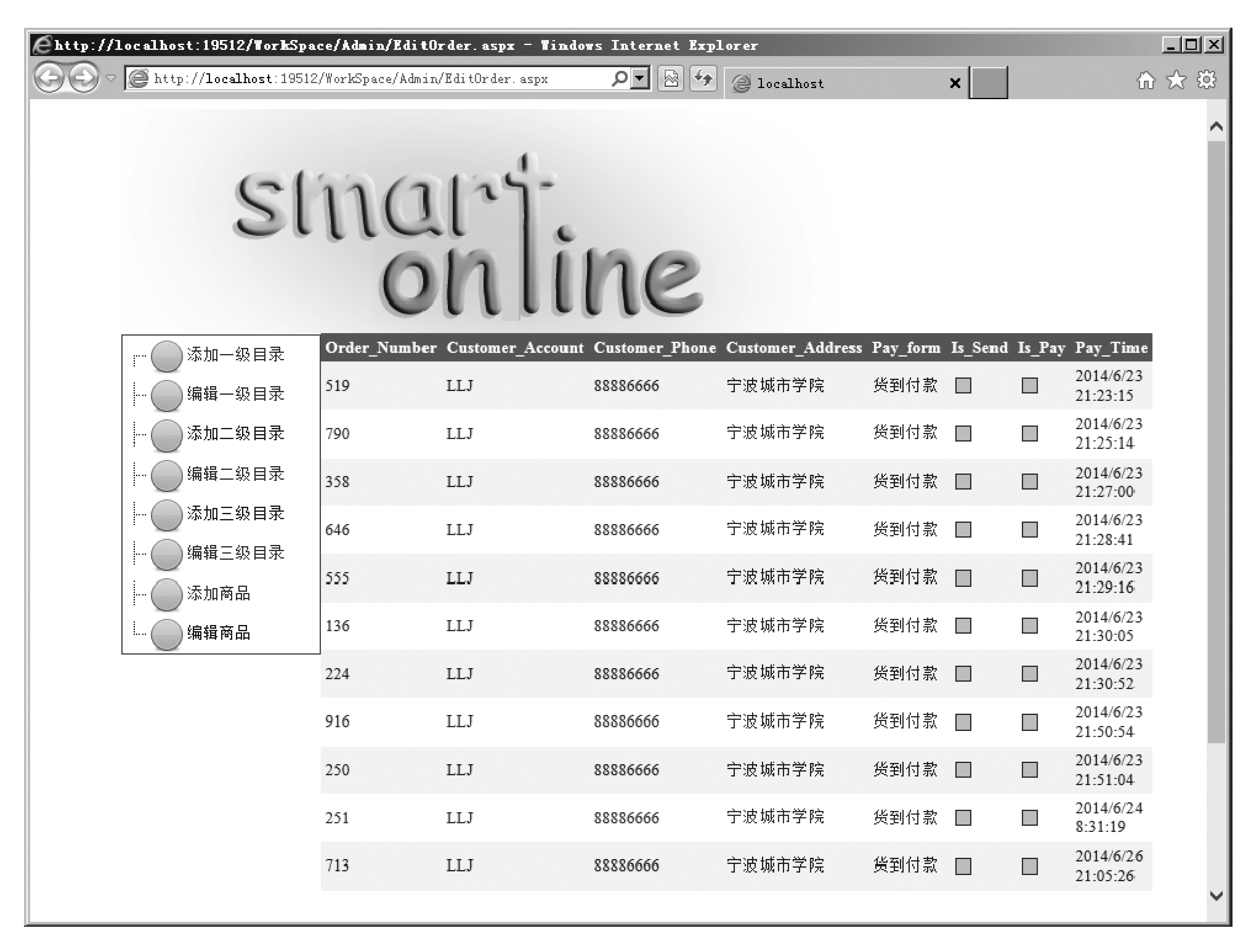
(3) 登录至 SQL Server 2008 服务器，单击“新建查询”按钮，打开“新建查询”窗口，输入下列 SQL 代码来创建 v_order 视图。

```
Create View v_order
AS
SELECT      T_Order.Order_ID,
    T_Order.Order_Number, T_Customer.Customer_Account,
    T_Customer.Customer_Phone, T_CustomerAddress.Customer_Address,
    T_Order.Pay_form, T_Order.Is_Send, T_Order.Is_Pay, T_Order.Pay_Time
    FROM  T_Order INNER JOIN  T_Customer
    ON T_Order.Customer_ID=T_Customer.Customer_ID INNER JOIN
    T_CustomerAddress ON T_Order.CustomerAddress_ID=
    T_CustomerAddress.CustomerAddress_ID AND
    T_Customer.Customer_ID=T_CustomerAddress.Customer_ID
```

(4) 拖动 GridView 控件到 EditOrder. aspx 页面中，单击智能提示，单击“自动套用格式”菜单项，单击“专业”列表项。按 F7 功能键，编写绑定 GridView 代码。按 F5 键运行结果，如图 8-13 所示。

```
public void Bind()
{
    ServiceReference1.ServiceSoapClient ssc=new
        ServiceReference1.ServiceSoapClient();
    GridView1.DataSource=ssc.QueryOperation("select * from v_order");
    GridView1.DataBind();
}
protected void Page_Load(object sender, EventArgs e)
{
    if(!Page.IsPostBack)
    {
        Bind();
    }
}
```

(5) 编辑 GridView 列。单击“智能提示”，单击“编辑列”菜单项。单击取消选中“自动生成字段”复选框。单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 Order_Number，单击 HeaderText 属性并输入“订单编号”。单击 ReadOnly 属性并将其设置为 true。单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 Customer_Account，单击 HeaderText 属性并输入“用户”。单击 ReadOnly 属性并将其设置为 true。单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 Customer_Email，单击 HeaderText 属性并输入“邮件”，单击 ReadOnly 属性并将其设置为 true。单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 Customer_address，单击 HeaderText 属性并输入“地址”，单击 ReadOnly 属性并将其设置为 true。单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 Pay_form，单击 HeaderText 属性并输入“支付方式”，



Order_Number	Customer_Account	Customer_Phone	Customer_Address	Pay_form	Is_Send	Is_Pay	Pay_Time
519	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:23:15
790	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:25:14
358	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:27:00
646	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:28:41
555	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:29:16
136	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:30:05
224	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:30:52
916	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:50:54
250	LLJ	88886666	宁波城市学院	货到付款			2014/6/23 21:51:04
251	LLJ	88886666	宁波城市学院	货到付款			2014/6/24 8:31:19
713	LLJ	88886666	宁波城市学院	货到付款			2014/6/26 21:05:26

图 8-13 显示订单数据

单击 ReadOnly 属性并将其设置为 true。单击 CheckBoxField 选项，单击“添加”按钮，单击 DataField 属性并输入 Is_Send，单击 HeaderText 属性并输入“是否发货”。单击 CheckBoxField 选项，单击“添加”按钮，单击 DataField 属性并输入 Is_Pay，单击 HeaderText 属性并输入“是否支付”。单击 BoundField 选项，单击“添加”按钮，单击 DataField 属性并输入 Pay_time，单击 HeaderText 属性并输入“订单时间”，单击 ReadOnly 属性并将其设置为 true。单击 CommandField 选项，单击“添加”按钮，单击“确认”按钮，完成效果如图 8-14 所示。

（6）实现“编辑”功能。单击 GridView1 控件，单击“事件”按钮，双击 RowEditing 事件，输入下列代码。

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
    GridView1.EditIndex=e.NewEditIndex;
    Bind();
}
```

（7）实现“取消”功能。单击 GridView1 控件，单击“事件”按钮，双击 RowCancelingEdit 事件，输入下列代码。

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
    GridView1.EditIndex=e.NewEditIndex;
    Bind();
}
```

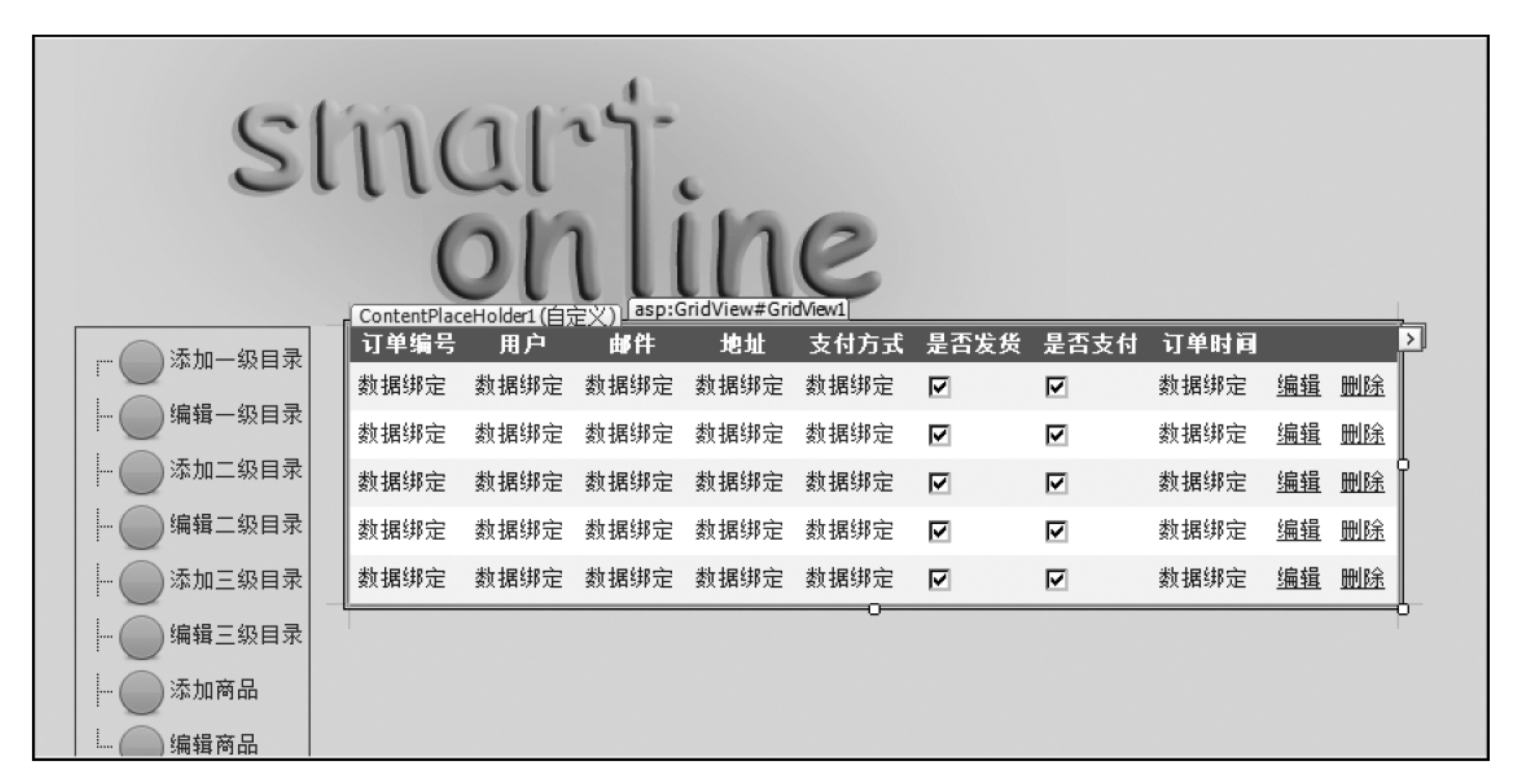


图 8-14　编辑列效果

(8) 实现"更新"功能。单击 GridView1 控件,单击 DataKeyNames 属性并输入 Order_ID,单击"事件"按钮,双击 RowUpdateing 事件。在 RowUpdateing 中不仅完成数据库的更新,而且还要向客户发送 E-mail 来通知客户相关订单状态的改变。

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    ServiceReference1.ServiceSoapClient ssc=
        new ServiceReference1.ServiceSoapClient();
    int is_s=Convert.ToInt32((GridView1.Rows[e.RowIndex].Cells[5].Controls[0] as
        CheckBox).Checked);
    int is_p=Convert.ToInt32((GridView1.Rows[e.RowIndex].Cells[5].Controls[0] as
        CheckBox).Checked);
    String order_id=GridView1.DataKeys[e.RowIndex][0].ToString();
    String sql=String.Format("update T_Order set is_send={0}, is_pay={1} where
        order_id={2}",
        is_s, is_p, order_id);
        ssc.ExeNonQuery(sql);
        GridView1.EditIndex=-1;
        Bind();
        //发送订单更新邮件
        String to=GridView1.Rows[e.RowIndex].Cells[2].Text;
        using(MailMessage mm=new MailMessage("lilijie51@gmail.com", to))
        {
            mm.Subject="订单状态变更";
            mm.Body="您的订单状态已经更新";
            mm.IsBodyHtml=false;
            SmtpClient smtp=new SmtpClient();
            smtp.Host="smtp.gmail.com";
            smtp.EnableSsl=true;
            NetworkCredential NetworkCred=new
                NetworkCredential("lilijie51@gmail.com", "111111");
```

```
            smtp.UseDefaultCredentials=true;
            smtp.Credentials=NetworkCred;
            smtp.Port=587;
            smtp.Send(mm);
            ClientScript.RegisterStartupScript(GetType(), "alert",
                "alert('Email sent.');", true);
        }
    }
```

按 F5 键运行程序，订单更新效果如图 8-15 所示。

smart online

添加一级目录
编辑一级目录
添加二级目录
编辑二级目录
添加三级目录
编辑三级目录
添加商品
编辑商品

订单编号	用户	邮件	地址	支付方式	是否发货	是否支付	订单时间		
519	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☑	☑	2014/6/23 21:23:15	编辑	删除
790	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☑	☐	2014/6/23 21:25:14	更新 取消	
358	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/23 21:27:00	编辑	删除
646	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/23 21:28:41	编辑	删除
555	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/23 21:29:16	编辑	删除
136	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/23 21:30:05	编辑	删除
224	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/23 21:30:52	编辑	删除
916	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/23 21:50:54	编辑	删除
250	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/23 21:51:04	编辑	删除
251	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/24 8:31:19	编辑	删除
713	LLJ	lilijie51@gmail.com	宁波城市学院	货到付款	☐	☐	2014/6/26 21:05:26	编辑	删除

图 8-15　订单更新效果

(9) 实现“删除”功能。单击“事件”按钮，双击 RowDeleting 事件，输入下列代码。

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    ServiceReference1.ServiceSoapClient ssc=
        new ServiceReference1.ServiceSoapClient();
    String order_id=GridView1.DataKeys[e.RowIndex][0].ToString();
    String sql=String.Format("delete from T_Order where order_id={0}",
      order_id);
    ssc.ExeNonQuery(sql);
    GridView1.EditIndex=-1;
    Bind();
}
```

8.5 总结归纳

本项目重点讲解了 Web Service 的开发和调用，以及采用 SMTP 协议实现订单更新效果的邮件通知。这两种技术在行业中应用得非常广泛，其优势之一就是系统集成。企业里经常要把用不同语言写成的、在不同平台上运行的各种程序集成起来，而这种集成将花费很大的开发力量。比如，应用程序经常需要从运行在 IBM 主机上的程序中获取数据，然后把数据发送到 UNIX 系统的应用程序中去。即使在同一个平台上，不同软件厂商生产的各种软件也常常需要集成起来。通过 Web Service，应用程序可以用标准的方法把功能和数据统一起来，以供其他应用程序使用。例如，有一个订单登录程序用于登录从客户端发来的新订单，包括客户信息、发货地址、数量、价格和付款方式等内容；还有一个订单执行程序，用于发送的实际货物的管理，这两个程序来自不同软件厂商。一份新订单进来之后，订单登录程序需要通知订单执行程序发送货物。通过在订单执行程序上面增加一层 Web Service 的功能，订单执行程序可以把 AddOrder 函数“暴露”出来。这样，每当有新订单到来时，订单登录程序就可以调用这个函数来发送货物了。

8.6 课后习题

简答题

1. Web 服务请求是否可以穿越防火墙？
2. 在 Web 服务的服务端产生的异常如何发送给客户端？
3. Web 服务系统的设计与面向对象的系统设计有何不同？
4. WebMethod 可以返回 DataSet 类型，那是否可以返回 DataTable 类型呢？为什么？

8.7 同步操练

格林酒店管理系统需要跟其他业务进行对接，需要为其他应用程序提供数据。项目经理要求开发人员采用 Web Service 方式将数据提供给其他业务部门，请开发该服务并提供服务地址。

项目 9　发布 Smart On Line 网站

9.1　项 目 引 入

Smart On Line 电子商城网站开发完毕并打算上线，需要将开发的电子商城网站发布到 IIS 中。

9.2　项 目 分 析

经过小组讨论和仔细分析，建议采用 IIS 作为 Web 服务器。在发布过程中需要考虑安全性，所以需要将 web.config 文件进行加密。

9.3　知 识 准 备

9.3.1　IIS 的安装和配置

Internet Information Services 7.0（IIS7）不仅仅是一个 Web 服务器，它更是一个安全性很强、易于管理的平台，适用于开发和可靠地寄存 Web 应用程序和服务。此外，IIS7 是 Windows Web 平台的主要增强，在统一的 Microsoft Web 平台技术——ASP.NET、Windows Communication Foundation Web 服务和 Windows SharePoint Services 中扮演着中心角色。若要体验 IIS7 的强大功能，请下载 Windows Server 2008 Release Candidate。IIS7 是 Microsoft 发布的迄今为止最强大的 Web 服务器，它提供了一组新功能，极大地改进了开发、部署和管理 Web 解决方案的方式。IIS7 的模块化设计使管理员拥有了前所未有的对其 Web 服务器进行控制的能力。灵活、可扩展的 IIS7 体系结构为开发人员自定义 Web 服务器提供了全新的机会。丰富的管理功能使得在 IIS7 上部署和管理 Web 应用程序比在任何其他 Web 服务器上更加简单和有效。最后，IIS7 强大的诊断和故障排除功能可以帮助用户快速鉴别并分类问题，极大地减少了停机时间。

1. Windows 7 中安装 IIS

单击“控制面板”→“程序和功能”中的“打开或关闭 Windows 功能”选项，如图 9-1 所示。接着会打开“Windows 功能”对话框。

单击“Windows 功能”对话框中“Internet 信息服务”前的＋号，展开相应节点，按照图 9-2 所示进行选择，然后单击“确定”按钮进行相应选项的安装，直至等待安装完毕。



图 9-1 “打开或关闭 Windows 功能”选项

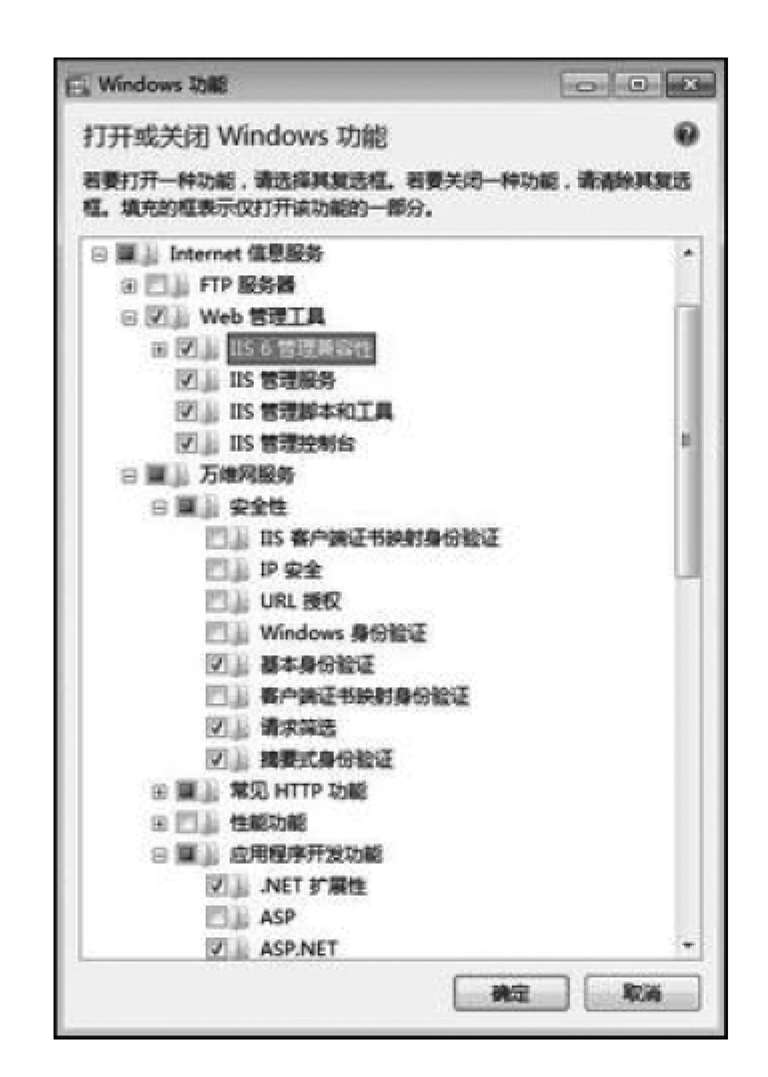


图 9-2 选择 IIS 安装选项

2. 注册 IIS

如果先安装 Visual Studio 2012，后安装 IIS，需要将 IIS 注册进 VS 中，具体方法为在“运行”命令对话框中输入 cmd，按 Enter 键进入 DOS 界面。输入下列命令：

```
cd C:\Windows\Microsoft.NET\Framework\v4.0.30319
```

按 Enter 键进入 C:\Windows\Microsoft.NET\Framework\v4.0.30319 这个文件夹目录下，输入 aspnet_regiis.exe -i 的命令，如图 9-3 所示。

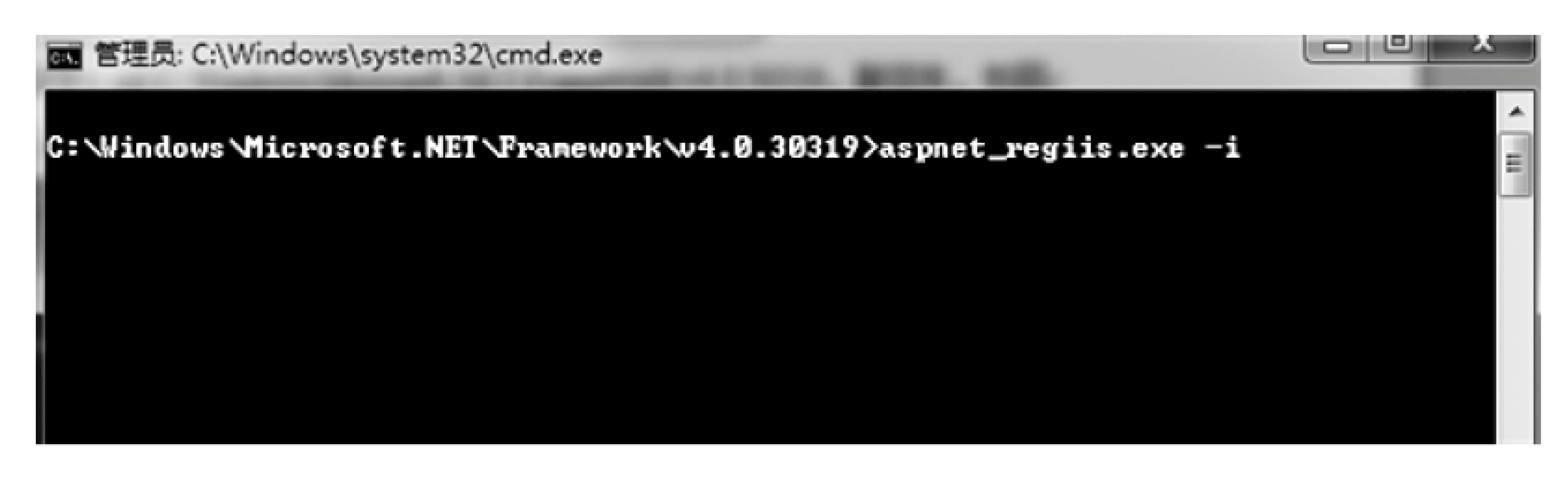


图 9-3 注册 IIS

9.3.2　加解密 web.config 配置文件

在.NET 中,一般把数据库连接字符串写在 web.config 这个文件里,即直接写在类中,在网站发布时,一般会把部署好的网站交给客户(包括页面文件和.dll 文件),没有代码文件(.cs 文件),如果把连接字符串写进代码里,显然很不方便。也有人会把连接字符串放一般程序中,这样在 web.config 里根本没有数据库连接信息。.NET 为开发人员提供了加密工具,只需要利用一个命令就可以加密 web.config 配置文件,所以在该文件中放置数据库连接字符串更安全。

1. 加密

进入 C:\Windows\Microsoft.net\Framework\v2.0.××××目录下,其中××××是所用的 Framework 版本,可以通过打开上述目录得到。输入如下命令:

```
aspnet_regiis -pef connectionStrings C:\Websites\BegAspNet2Db
```

注意: C:\Websites\BegAspNet2Db 并不是一个真正存在的目录,应用时要用站点的根目录替代它。当再次打开 Web.config 配置文件时,会发现 connectionStrings 中已经不再具有任何可以获得的有效信息,取而代之的是一些杂乱的字符。

2. 解密

如果要修改加密文件中的某些信息,可以对加密的内容进行解密。命令如下:

```
aspnet_regiis -pdf connectionStrings C:\Websites\BegAspNet2D
```

9.4　项 目 实 施

9.4.1　任务1: 加密 web.config 配置文件

1. 任务目标

能使用命令对 web.config 文件进行加密。

2. 任务内容

对 Smart On Line 网站中的 web.config 文件进行加密。

3. 任务实施步骤

(1) 生成发布文件。单击"生成"→"发布"菜单项,打开"发布网站"对话框,选择 C 盘下的 SmartOnLine 文件夹,如图 9-4 所示,单击"确定"按钮。

(2) 单击"开始"→"附件"→"命令提示符"菜单项,打开"命令提示符"窗口。输入 cd C:\Windows\Microsoft.NET\Framework64\v4.0.30319。按 Enter 键。再输入 aspnet_regiis -pef connectionStrings C:\SmartOnLine,按 Enter 键进行加密操作,如图 9-5 所示。

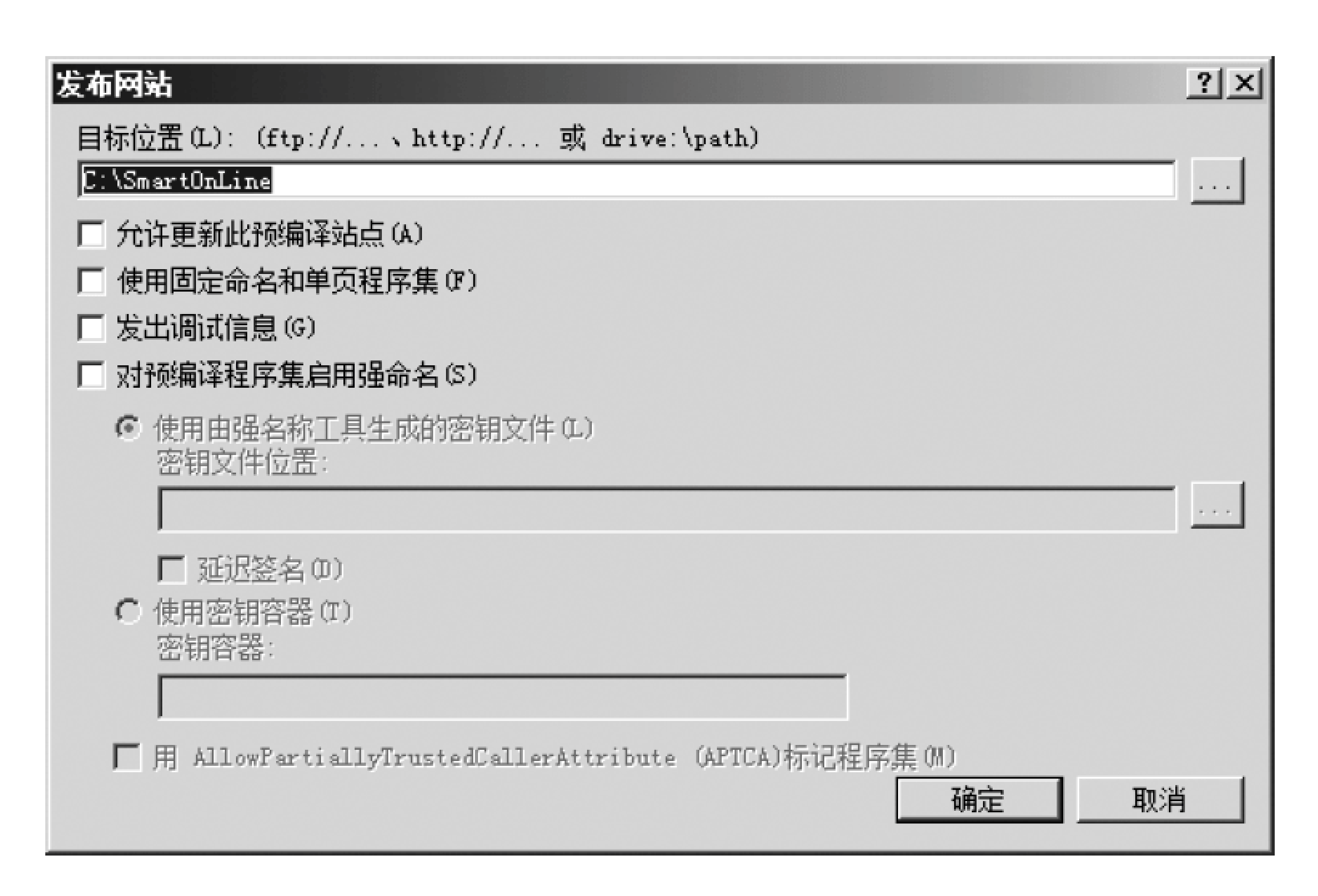


图 9-4 “发布网站”对话框

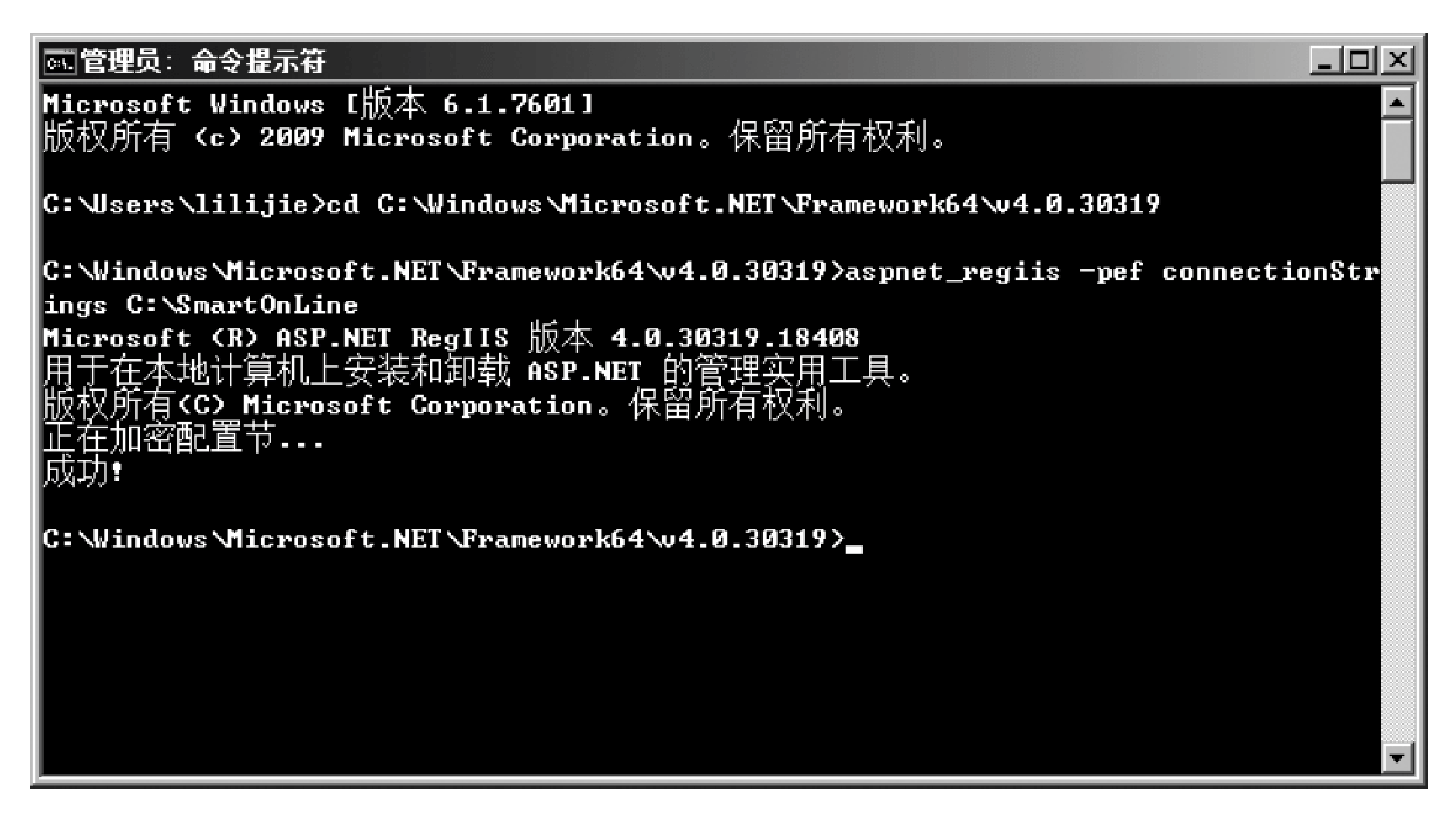


图 9-5 加密 web.config 文件

9.4.2 任务 2：发布 Smart On Line 网站

1. 任务目标

能在 IIS 环境中发布 ASP.NET 网站。

2. 任务内容

发布 Smart On Line 网站。

3. 任务实施步骤

(1) 双击“控制面板”→“管理工具”→“Internet 信息服务(IIS)管理器”图标，打开 IIS 管理器，如图 9-6 所示。

(2) 右击左边的“网站”文件夹，单击“添加网站”菜单项，打开如图 9-7 所示的“添加网站”对话框，输入网站的相应属性：名称、物理路径、端口号(与其他应用程序端口号不要重复)，单击“确定”按钮。



图 9-6　IIS 管理器

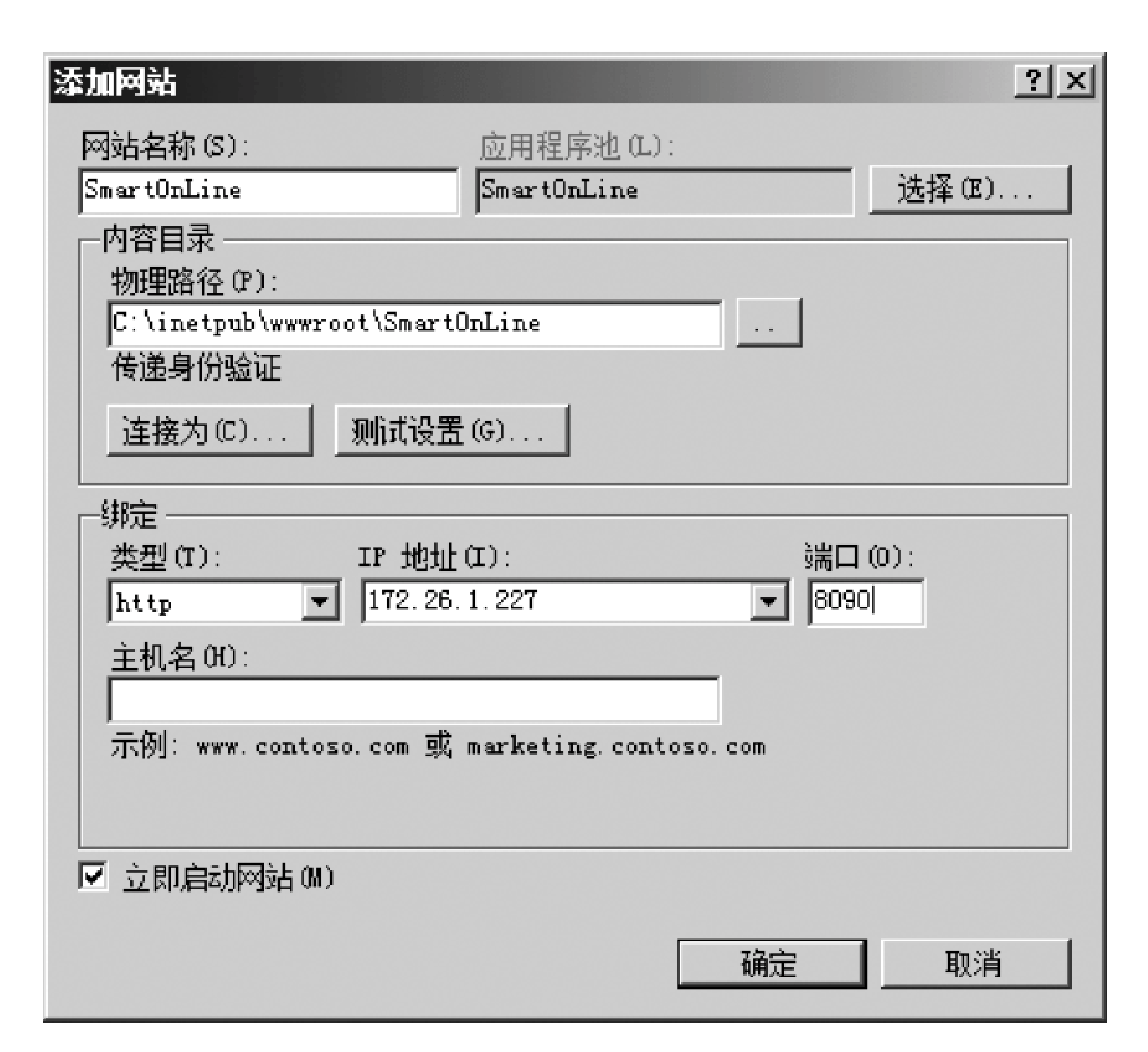


图 9-7　“添加网站”对话框

(3) 双击 IIS 管理器中的“应用程序池”选项，双击“SmartOnLine”程序池，在打开的对话框中单击.NET Framework v4.0.30319，单击“确定”按钮，如图 9-8 所示。因为机器上有可能安装了若干版本的.Net Framework，所以需要选择对应的版本。

(4) 打开浏览器，在地址栏中输入 http://172.26.1.227:8090/Shop/index.html，按 Enter 键即可看到网站的首页，如图 9-9 所示。



图 9-8　配置应用程序池



图 9-9　Smart On Line 网站发布后的效果

9.5　总结归纳

发布网站这个子项目任务难度不高，主要要求读者掌握 web.config 文件的加、解密和网站发布的方法。发布过程中务必要选择正确的.NET 框架版本号，否则会出现发布错误。

9.6　课后习题

编程题

将任务 1 中加密的 web.config 解密。

9.7　同步操练

现已完成格林酒店管理系统的开发，项目经理要求开发人员将该系统发布到网站上。

参 考 文 献

[1] 段菲，刘宝弟，陈正华. ASP.NET 应用程序开发[M]. 北京：清华大学出版社，2013.
[2] 郑阿奇，刘启芬，顾韵华. SQL Server 数据库教程[M]. 北京：人民邮电出版社，2012.
[3] 徐阳，丁小峰. Head First HTML 与 CSS[M]. 第 2 版. 北京：中国电力出版社，2013.
[4] 淘宝前端团队. JavaScript 权威指南[M]. 第 6 版. 北京：机械工业出版社，2012.
[5] 贾洪峰. ADO.NET 4 从入门到精通[M]. 北京：清华大学出版社，2012.
[6] 胡德华. SOA 之道——思想、技术、过程与实践[M]. 上海：上海交通大学出版社，2011.
[7] 史岩. ASP.NET 批量发送邮件[J]. 电脑编程技巧与维护，2011(13)：53-58.
[8] 颜炯. IIS 7 开发与管理完全参考手册[M]. 北京：清华大学出版社，2009.